A multi-objective stochastic programming model for disaster relief logistics under uncertainty

Helsa Heishun Chan

August 8, 2015

Abstract

This report presents an application of a multi-objective stochastic programming model for disaster relief logistics. We revise the mathematical model proposed by Bozorgi-Amiri et al. (2013) and develop a program, which adopts SYMPHONY as our mixed-integer linear programming solver, to solve the optimization problem. Our program can also be run in parallel to speed up computation. In this report, we also conduct a case study of a disaster relief planning problem in Iran to demonstrate its practicality and a computational study performance of our approach.

1 Introduction

Every year, natural disasters including earthquakes, drought, flood and tropical cyclone kill thousands of people and cause large-scale destruction to habitats which often results in a loss of millions of dollars of assets. A well-designed disaster relief management system aids affected people and allows reconstruction at the affected areas in the most efficient way, while an inappropriate allocation of resources results in higher unnecessary costs, supply shortage and increased suffering. Thus, a sophisticated disaster relief management system developed in the pre-disaster stage is highly useful in reducing costs and maximizing efficiency in saving lives.

However, the huge complexity and dynamics involved in disasters naturally implies a high level of uncertainty in disaster relief management (Bozorgi-Amiri et al., 2013). Two main features of the problems that a disaster relief planner would face, outlined by Bozorgi-Amiri et al., include conflicting objectives, such as minimizing costs while maximizing satisfaction in affected areas, and the lack of knowledge of data, such as demand, supply and cost. Bozorgi-Amiri et al. (2013) proposed a stochastic programming model which aims to "model disaster planning and response capturing the inherent uncertainty in demand, supply, and cost resulting from a disaster" (Bozorgi-Amiri et al., 2013).

Based on the research carried out by Bozorgi-Amiri et al. (2013), this paper aims to enhance the model with greater flexibility and efficiency. Like the model by Bozorgi-Amiri et al. (2013), we first formulate our model as a linear programming problem. Then, we build a C program, which integrates a mixed-integer linear programming solver SYMPHONY (Ralphs et al., 2015), to find an optimal solution of various case studies. Finally, we use multiprocessing to speed up the process and solve cases in larger scale.

1.1 Previous Results

Bozorgi-Amiri et al. (2013) proposed a multi-objective robust stochastic programming approach based on two objectives in disaster relief logistics. The first objective is to minimize costs on the pre-disaster

setup including inventory procurement and transportation as well as the post-disaster transportation, holding and shortage; to give a robust model, cost variability and penalty of infeasibility are also taken into consideration. The second objective is to maximize the affected area's overall satisfaction, which is achieved by minimizing the sum of their maximum shortage. The two objectives are formulated and combined into one objective function, so that a compromise solution can be obtained by seeking an optimal feasible solution, and by changing the weighting, a weighted compromise solution can be achieved.

This approach assumed that the disaster relief logistics network consists of three types of parties, namely the set of suppliers, relief distribution centers (RDCs) and affected areas (AAs). Four types of flow of commodities were considered: pre-disaster flow from a supplier to an RDC, post-disaster flow from a supplier to an RDC, post-disaster flow from an RDC to another RDC, and post-disaster flow from an RDC to an AA. In this model, the set of AAs and candidate RDCs were considered the same and occasionally denoted as nodes, and at each demand node, an RDC could be set up when necessary to store, receive or send commodities. Within the same location, suppliers and RDCs were positioned "close" to each other and the AAs to minimize the distance for goods distribution. Other assumptions include the possibility of disruption of capability of suppliers and RDCs, the uncertainty of demand for AAs and cost depending on different disaster situations, and that three types of storage capacity for each RDC (small, medium or large) are allowed. Each case of disaster situation is called a scenario. Figure 1 shows two simplified flow charts of the same case under different scenarios, each with 4 nodes for AAs and candidate RDCs, where suppliers are present on 3 of the nodes. An arrow indicates a flow of commodities and different colors of arrows represent different types of flow.

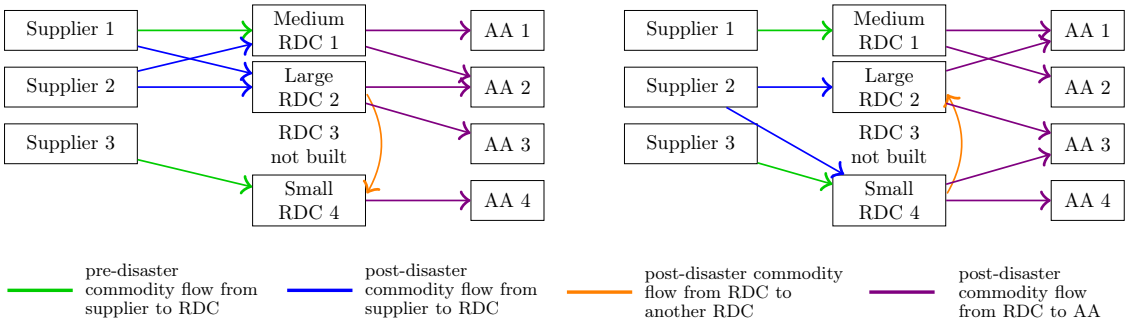

Figure 1: General schema of relief distribution chain of the model (Bozorgi-Amiri et al. 2013)

Notice that the pre-disaster flows from suppliers to RDCs are independent of scenarios, while the post-disaster flows depend on scenarios.

The model was implemented for a case study in a region at Iran where five suppliers and fifteen demand points were considered. Three types of commodities and four scenarios were considered. Together with assumptions including setup costs, transportation costs, unit commodity costs and occurrence probability of each scenario, the model is formulated to seek the optimal plan. The case study and results are found in Sections 3.4 and 3.5.

2 System Overview

2.1 SYMPHONY

SYMPHONY is an open-source solver developed by the Computational Infrastructure for Operations Research (COIN-OR) for solving mixed-integer linear programs (MILP) using the branch-and-cut method. SYMPHONY can be run as an interactive window, which lets users load a linear programming problem and displays the solution, or as a callable library in C, which allows users to develop their own application for solving a specific MILP. In addition, SYMPHONY allows the implementation of parallel programming which is extremely useful in shortening run-time and increasing efficiency.

2.2 The Program

In this study, we use the callable library feature to build a C program to seek an optimal plan in disaster relief planning. The program runs as the following:

1. Input files are read by the program and the data (i.e. problem parameters) are allocated into arrays.
2. Each variable is assigned an index indicating its column position in the constraint matrix.
3. Coefficient of each term in the objective function is assigned, and bounds for each term are specified. This sets up the objective function into use.
4. Each variable is specified its nature regarding whether or not it has to be an integer.
5. The problem is created and loaded onto SYMPHONY using `sym_explicit_load_problem`.
6. Constraints are added into the problem by assigning coefficients to the columns corresponding to the relating variables. `sym_add_row` is called to add a row into the constraint matrix.
7. `sym_solve` solves the linear program; `sym_get_col_solution` and `sym_get_obj_val` are called to display the optimal solution and the optimal value obtained by SYMPHONY.
8. The results are printed into files of csv format for future use.

Features of SYMPHONY including limits and warm start are occasionally set so that an intermediate solution can be outputted or loaded. `sym_set_int_param` is called to change the values of `time_limit` or `gap_limit` to set a time limit or optimality gap limit, meaning that once a limit is reached, SYMPHONY can output the intermediate solution stored. The warm start feature allows us to load an intermediate solution to the program to save the effort and time solving it from scratch. If we wish to use the warm start feature, we first need to call `sym_write_warm_start_desc` in the previous run to produce an output file containing the intermediate solution. Then, we call `sym_read_warm_start` to read this output file and load the solution and use `sym_warm_solve` to perform warm solve.

SYMPHONY is known to be compatible with parallel processing. Since this case study is relatively small in magnitude, the runtime is as short as 0.9 seconds, yet for larger cases it could take a few hours or even longer than a day. Methods in improving the speed of the solver is worth investigation. Details on the performance of SYMPHONY in solving large-size case studies are covered in Section 3.6.

3 Our Work

3.1 Overview

This model follows most assumptions of the model by Bozorgi-Amiri et al. (2013) except for the following two points:

First, linear constraints are directly used in our formulation to save the effort of rewriting non-linear terms into linear terms, hence simplifies the program and allows faster computation. This is especially important in the case study, where much higher number of locations and scenarios are considered. Specifically, the two objective functions and constraints (24)–(27) in the original model by Bozorgi-Amiri et al. (2013) are modified to suit our needs. For convenience and clarity, an extra variable is introduced to indicate flows of commodities between two RDCs.

Second, RDCs and AAs with the same indices no longer represent the same location. In addition, suppliers and RDCs are no longer intentionally placed close to each other or close to the AAs. This way more flexibility and possibilities are provided in seeking the best positions for RDCs to be set up. If we consider a randomly generated case with 5 suppliers, 6 RDCs, 10 AAs and 3 possible sizes of RDCs on a map, the following is a possible graph displaying the flow of commodities:

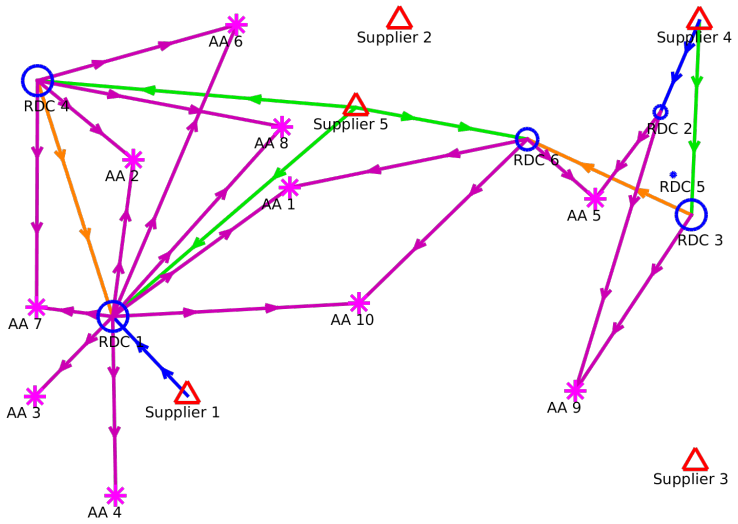

Figure 2: Example of relief distribution chain in our model under a certain scenario

Green lines indicate a pre-position of inventory and blue lines represent the post-disaster flow of commodities from suppliers (red triangles) to RDCs (blue circles). Orange lines show the start-end positions of RDC-RDC flows. Finally, purple lines indicate flows of commodities from RDCs to AAs (magenta asterisks). An absence of connection between two nodes means there is no transfer of commodities between them. The four different sizes of RDCs, from large to small, indicate a large size, medium size, small size, and absence of RDCs respectively. For instance, RDC 3 is large, RDC 6 is medium and no RDC is built at candidate RDC node 5.

3.2 Model Formulation

This is the formulation of our model:

Sets / Indices

I	Set of suppliers indexed by $i \in I$
J	Set of candidate RDCs indexed by $j \in J$
K	Set of AAs indexed by $k \in K$
L	Set of size of RDCs indexed by $l \in L$
S	Set of possible scenarios indexed by $s \in S$
C	Set of commodities indexed by $c \in C$

Parameters

p_s	Occurrence probability of scenario s
F_{jl}	Fixed cost for opening a RDC of size l at location j
φ_{ic}	Procuring cost of a unit commodity c from supplier i
φ_{ics}	Procuring cost of a unit commodity c from supplier i in scenario s
C_{ijc}	Transportation cost for a unit commodity c from supplier i to RDC j
C_{ijcs}	Transportation cost for a unit commodity c from supplier i to RDC j in scenario s
$C_{j_1j_2cs}$	Transportation cost for a unit commodity c from RDC j_1 to RDC j_2 in scenario s
C_{jkcs}	Transportation cost for a unit commodity c from RDC j to AA k in scenario s
h_{kc}	Inventory holding cost for a unit commodity c at AA k
π_c	Shortage cost for a unit commodity c
v_c	Required unit space for commodity c
D_{kcs}	Amount of demand for commodity c at AA k in scenario s
S_{ic}	Amount of commodity c that can be supplied from supplier i
Cap_l	Capacity of RDC of capacity category l
ρ_{ics}	Fraction of stocked material of commodity c at supplier i that remains usable in scenario s $(0 \leq \rho_{ics} \leq 1)$
ρ_{jcs}	Fraction of stocked material of commodity c at RDC j that remains usable in scenario s $(0 \leq \rho_{jcs} \leq 1)$
λ_1, λ_2	Weight assigned to cost variability
M	A very large number

Continuous and binary decision variables

Q_{ijc}	Amount of commodity c procured from supplier i and stored at the RDC j
X_{ijcs}	Amount of commodity c transferred from supplier i to RDC j in scenario s
$Y_{j_1j_2cs}$	Amount of commodity c transferred from RDC j_1 to RDC j_2 in scenario s
Z_{jkcs}	Amount of commodity c transferred from RDC j to AA k in scenario s
I_{kcs}	Amount of inventory of commodity c held at AA k in scenario s
b_{kcs}	Amount of shortage of commodity c at AA k in scenario s
δ_{jl}	1 if RDC with capacity category l is located at candidate RDC j; 0 otherwise

3.3 Mathematical Formulation

(1) to (8) are defined for the convenience of formulation:

Pre-disaster phase:

$$\sum_{j\in J}\sum_{l\in L} F_{jl}\cdot\delta_{jl} \qquad \text{SC (Setup costs)} \tag{1}$$

$$\sum_{i\in I}\sum_{j\in J}\sum_{c\in C} \varphi_{ic}\cdot Q_{ijc} \qquad \text{PC (Procuring costs)} \tag{2}$$

$$\sum_{i\in I}\sum_{j\in J}\sum_{c\in C} C_{ijc}\cdot Q_{ijc} \qquad \text{TC (Transportation costs from suppliers to RDCs)} \tag{3}$$

Post-disaster phase:

$$\sum_{i\in I}\sum_{j\in J}\sum_{c\in C} \varphi_{ics}\cdot X_{ijcs} \qquad \text{PC}_s \text{ (Procuring costs)} \tag{4}$$

$$\sum_{i\in I}\sum_{j\in J}\sum_{c\in C} C_{ijcs}\cdot X_{ijcs} \qquad \text{TCX}_s \text{ (Transportation costs from suppliers to RDCs)} \tag{5}$$

$$\sum_{j_1\in J}\sum_{j_2\in J}\sum_{c\in C} C_{j_1j_2cs}\cdot Y_{j_1j_2cs} \qquad \text{TCY}_s \text{ (Transportation costs from RDCs to RDCs)} \tag{6}$$

$$\sum_{j\in J}\sum_{k\in K}\sum_{c\in C} C_{jkcs}\cdot Z_{jkcs} \qquad \text{TCZ}_s \text{ (Transportation costs from RDCs to AA)} \tag{7}$$

$$\sum_{k\in K}\sum_{c\in C} h_{kc}\cdot I_{kcs} \qquad \text{IC}_s \text{ (Inventory holding costs in AAs)} \tag{8}$$

$$\sum_{k\in K}\sum_{c\in C} \pi_{c}\cdot b_{kcs} \qquad \text{SC}_s \text{ (Shortage costs in AAs)} \tag{9}$$

In addition we let

$$\text{PRE} = \text{SC} + \text{PC} + \text{TC} \text{ (Total pre-disaster costs)} \tag{10}$$

$$\text{POST}_s = \text{PC}_s + \text{TCX}_s + \text{TCY}_s + \text{TCZ}_s + \text{IC}_s + \text{SC}_s \text{ (Total post-disaster costs in scenario } s) \tag{11}$$

Based on the formulation by Bozorgi-Amiri et al. (2013) the above discussion is modified and formulated as follows:

$$\text{Min Obj}_1 = \text{PRE} + \sum_{s\in S} p_s(\text{POST}_s) + \lambda_1 \sum_{s\in S} p_s \left[\left(\text{POST}_s - \sum_{s'\in S} p_{s'}(\text{POST}_{s'})\right) + 2\theta_{1s}\right] \tag{12}$$

$$\begin{aligned}\text{Min Obj}_2 = &\sum_{s\in S} p_s \left(\sum_{c\in C} \max_{k\in K}\{b_{kcs}\}\right) \\ &+ \lambda_2 \sum_{s\in S} p_s \left[\left(\sum_{c\in C} \max_{k\in K}\{b_{kcs}\} - \sum_{s'\in S} p_{s'} \sum_{c\in C} \max_{k\in K}\{b_{kcs'}\}\right) + 2\theta_{2s}\right]\end{aligned} \tag{13}$$

subject to

$$\sum_{i\in I} X_{ijcs} + \rho_{jcs} \cdot \sum_{i\in I} Q_{ijc} + \sum_{j'\in J\setminus\{j\}} Y_{j'jcs} = \sum_{j'\in J\setminus\{j\}} Y_{jj'cs} + \sum_{k\in K} Z_{jkcs}, \forall\, j \in J, c \in C, s \in S \tag{14}$$

$$\sum_{j\in J} Z_{jkcs} - D_{kcs} = I_{kcs} - b_{kcs}, \forall\, k \in K, c \in C, s \in S \tag{15}$$

$$\sum_{i\in I}\sum_{c\in C} X_{ijcs} \le M \cdot \sum_{l\in L} \delta_{jl}, \forall\, j \in J, s \in S \tag{16}$$

$$\sum_{j_2\in J}\sum_{c\in C} Y_{j_1 j_2 cs} \le M \cdot \sum_{l\in L} \delta_{j_1 l}, \forall\, j_1 \in J, s \in S \tag{17}$$

$$\sum_{j_1\in J}\sum_{c\in C} Y_{j_1 j_2 cs} \le M \cdot \sum_{l\in L} \delta_{j_2 l}, \forall\, j_2 \in J, s \in S \tag{18}$$

$$\sum_{k\in K}\sum_{c\in C} Z_{jkcs} \le M \cdot \sum_{l\in L} \delta_{jl}, \forall\, j \in J, s \in S \tag{19}$$

$$\sum_{i\in I}\sum_{c\in C} v_c \cdot Q_{ijc} \le \sum_{l\in L} \mathrm{Cap}_l \cdot \delta_{jl}, \forall\, j \in J \tag{20}$$

$$\sum_{j\in J} Q_{ijc} \le S_{ic}, \forall\, i \in I, c \in C \tag{21}$$

$$\sum_{j\in J} X_{ijcs} \le \rho_{ics} \cdot S_{ic}, \forall\, i \in I, c \in C, s \in S \tag{22}$$

$$\sum_{l\in L} \delta_{jl} \le 1, \forall\, j \in J \tag{23}$$

$$\mathrm{POST}_s - \sum_{s'\in S} p_{s'}(\mathrm{POST}_{s'}) + \theta_{1s} \ge 0, \forall\, s \in S \tag{24}$$

$$\sum_{c\in C} \max_{k\in K}\{b_{kcs}\} - \sum_{s'\in S} p_{s'} \cdot \left(\sum_{c\in C} \max_{k\in K}\{b_{kcs'}\}\right) + \theta_{2s} \ge 0, \forall\, s \in S \tag{25}$$

$$\delta_{jl} \in \{0,1\}, Q_{ijc}, X_{ijcs}, Y_{j_1 j_2 cs}, Z_{jkcs}, I_{kcs}, b_{kcs}, \theta_{1s}, \theta_{2s} \ge 0$$
$$\forall\, i \in I, j, j_1, j_2 \in J, k \in K, l \in L, c \in C, s \in S \tag{26}$$

Objective Function 1 minimizes the expected value of the total cost. The first term gives the expected value of pre-disaster total cost and the expected response phase costs. The second term is the cost variability, with the absolute deviation proposed by Yu and Li (2000). Objective Function 2 minimizes the expected sum of maximum shortage. The first term gives the expected maximal shortage and the second term measures the shortage variability using model in Yu and Li (2000).

The first constraint, Equation 14, is designed for commodity flow balance. Ideally, if an RDC is built at location j, Equation 14 balances the inflow and outflow of each commodity at each RDC in each scenario, so that no overflow or shortage occurs at that RDC. Otherwise, all terms in Equation 14 would be taken care by Equations 16–20 and equal to zero.

Equation 15 gives the amount of each commodity at each AA in each scenario. A positive value on both sides of Equation 15 indicates an overflow, while a negative value indicates a shortage. As an

overflow and a shortage are penalized in the objective functions, both sides are minimized as small as possible in an optimal solution.

Inequalities 16–23 are included for model feasibility. Inequalities 16–19 ensure no commodities are delivered to or from some candidate RDC at location j in each scenario if no RDC is built at that location. Inequality 20 ensures the amount of each commodity delivered to each RDC does not exceed its capacity for that type of commodity in each scenario. If no RDC is built at a location, this equation makes sure no commodity is stored at that location. Then, Inequalities 21 and 22 bound the amount of each commodity sent out by each supplier before and after a disaster so as to not exceed the possible amount that can be supplied by that supplier respectively, and Inequalities 23 assures at most one RDC is built at each candidate RDC location.

Finally, Inequalities 24–25 specify the range of θ_{1s} and θ_{2s} required in the model of Yu and Li (2000) for variability calculation.

3.4 Case Study

We begin by reproducing the case study results of Iran by Bozorgi-Amiri et al. (2013) using our model:

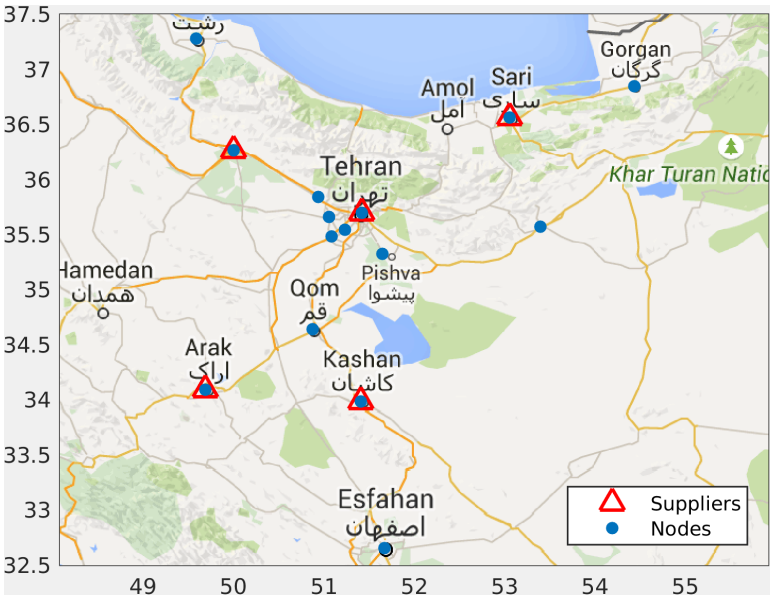

Figure 3: Map of case study: nodes and suppliers

Here we consider a case of five suppliers and 15 nodes in a region of Iran. RDCs and AAs are coincidentally the same set of locations as represented by the 15 nodes, and it is assumed that suppliers are built at five of these nodes. Three types of commodities: water, food and shelter, and four scenarios (s_1, s_2, s_3, s_4), with occurrence probabilities 0.45, 0.3, 0.1 and 0.15, are taken into consideration.

The locations corresponding the nodes are shown in Table 1, where suppliers are labeled with the triangle symbol, and the capacity of each supplier for each commodity (S) is shown in Table 2. Table 3 and Table 4 give $F, \text{Cap}_l, \varphi, v$ and the unit transportation costs.

Location	Latitude	Longitude
Gorgan, Golestan (GO)	36.845643N	54.439336E
Semnan, Semnan (SM)	35.572778N	53.397222E
△ Sari, Mazadaran (SA)	36.563333N	53.056000E
Rasht, Gilan (RS)	37.280833N	49.583056E
△ Qazvin, Qazvin (QZ)	36.266667N	50.000000E
Karaj, Alborz (KR)	35.840019N	50.939091E
△ Tehran, Tehran (TE)	35.696111N	51.423056E
Varamin, Tehran (VA)	35.325241N	51.647199E
Robat Karim, Tehran (RK)	35.484722N	51.082778E
Eslamshahr, Tehran (ES)	35.544581N	51.230246E
Shahriar, Tehran (SH)	35.659722N	51.059167E
Qom, Qom (QM)	34.639944N	50.875942E
△ Arak, Markazi (AR)	34.091667N	49.689167E
△ Isfahan, Isfahan (IS)	32.654628N	51.667983E
Kashan, Isfahan (KS)	33.985036N	51.409963E

Table 1: Locations of nodes and suppliers

Suppliers	(Water, food, shelter)
Sari	(450, 450, 150)
Qazvin	(450, 450, 150)
Tehran	(510, 510, 170)
Arak	(450, 450, 150)
Isfahan	(450, 450, 150)

Table 2: Capacity of suppliers for each commodity (10^3 units)

Size	$F(10^3\$)$	Cap(10^3 m^3)
Small	500	10
Medium	800	16
Large	1,200	24

Table 3: Facility setup cost and capacity of each available size of RDCs

Commodity	φ (\$)	v (m^3/unit)	Transport (\$/unit-km)
Water	0.5	0.0045	0.6
Food	2	0.002	0.15
Shelter	20	0.12	1.8

Table 4: Unit procure price, volume occupied by commodity and transportation cost

Notice we assume φ to vary only with the commodity in this case. Holding costs (h) are estimated by the current procurement prices of commodities (φ) and unmet demand penalties (π) are assumed to be ten times that of φ. Demand under each scenario is shown in Table 5:

Node	Gorgan	Semnan	Sari	Rasht	Qazvin
s_1	(319, 319, 106)	(34, 34, 11)	(579, 579, 193)	(524, 524, 175)	(68, 68, 23)
s_2	(159, 159, 53)	(69, 69, 23)	(521, 521, 174)	(429, 429, 143)	(169, 169, 56)
s_3	(96, 96, 32)	(29, 29, 10)	(289, 289, 96)	(238, 238, 79)	(158, 158, 53)
s_4	(287, 287, 96)	(57, 57, 19)	(579, 579, 193)	(476, 476, 159)	(113, 113, 38)
Node	Karaj	Tehran	Varamin	Robat Karim	Eslamshahr
s_1	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
s_2	(293, 293, 98)	(1418, 1418, 473)	(103, 103, 34)	(138, 138, 46)	(98, 98, 33)
s_3	(222, 222, 74)	(1654, 1654, 551)	(81, 81, 27)	(100, 100, 33)	(76, 76, 25)
s_4	(256, 256, 85)	(1339, 1339, 446)	(76, 76, 25)	(94, 94, 31)	(67, 67, 22)
Node	Shahriar	Qom	Arak	Isfahan	Kashan
s_1	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)	(0, 0, 0)
s_2	(188, 188, 63)	(228, 228, 76)	(22, 22, 7)	(225, 225, 75)	(30, 30, 10)
s_3	(177, 177, 59)	(187, 187, 62)	(75, 75, 25)	(990, 990, 330)	(30, 30, 10)
s_4	(146, 146, 49)	(166, 166, 55)	(22, 22, 7)	(225, 225, 75)	(21, 21, 7)

Table 5: Demand (D_{kcs}) in 10^3 units for water, food and shelter

Transportation costs (C) between nodes are estimated by distance and are assumed to be fixed among scenarios. Post-disaster costs are assumed to be 1.8 times that of the pre-disaster phase. Table 6 shows a distance chart with reference to the Google distance API in kilometers:

From\To	GO	SM	SA	RS	QZ	KR	TE	VA	RK	ES	SH	QM	AR	IS	KS
GO	0	303	134	500	557	459	412	429	457	434	447	554	683	854	650
SM	326	0	198	552	373	275	220	198	248	239	271	283	417	584	379
SA	134	197	0	364	421	373	275	293	321	298	311	417	547	718	514
RS	499	550	364	0	174	283	332	384	328	337	304	445	487	641	564
QZ	557	372	421	176	0	105	154	206	149	158	126	272	315	469	332
KR	460	275	324	279	100	0	57.1	109	52.4	61.3	29.2	192	310	464	289
TE	409	220	274	330	152	53.7	0	57.5	62	28.4	43.1	157	288	458	254
VA	424	192	288	379	200	102	52.4	0	78.1	61	100	116	250	417	212
RK	454	254	318	327	148	50.1	53.4	77.8	0	17.4	22.6	123	240	39.4	219
ES	433	236	298	33.7	159	60.8	28.9	62.5	16.8	0	27.5	131	262	43.1	227
SH	450	253	315	304	126	27.7	41.7	99.9	24.2	27.8	0	145	258	412	241
QM	551	289	416	437	264	193	151	123	123	134	149	0	135	315	111
AR	681	425	546	492	318	310	281	260	246	265	261	135	0	331	248
IS	849	575	713	645	471	463	450	409	423	434	414	273	333	0	216
KS	645	371	510	566	339	289	247	205	220	231	245	104	247	218	0

Table 6: Distance matrix (km)

Finally, Table 7 shows the percentage of commodity that remains usable after a disaster at each node (ρ_{jcs}):

Node	Gorgan	Semnan	Sari	Rasht	Qazvin
s_1	(80, 77, 85)	(93, 90, 98)	(75, 72, 80)	(82, 79, 87)	(100, 100, 100)
s_2	(90, 87, 95)	(88, 85, 93)	(82, 79, 87)	(82, 79, 87)	(85, 82, 90)
s_3	(94, 91, 99)	(95, 92, 100)	(90, 87, 95)	(92, 89, 97)	(87, 84, 92)
s_4	(82, 79, 87)	(90, 87, 95)	(80, 77, 85)	(81, 78, 86)	(90, 87, 95)
Node	Karaj	Tehran	Varamin	Robat Karim	Eslamshahr
s_1	(100, 100, 100)	(100, 100, 100)	(100, 100, 100)	(100, 100, 100)	(100, 100, 100)
s_2	(83, 80, 88)	(82, 79, 87)	(81, 78, 86)	(78, 75, 83)	(78, 75, 83)
s_3	(87, 84, 92)	(79, 76, 84)	(85, 82, 90)	(84, 81, 89)	(83, 80, 89)
s_4	(85, 82, 90)	(83, 80, 88)	(86, 83, 91)	(85, 82, 90)	(85, 80, 90)
Node	Shahriar	Qom	Arak	Isfahan	Kashan
s_1	(100, 100, 100)	(100, 100, 100)	(100, 100, 100)	(100, 100, 100)	(100, 100, 100)
s_2	(82, 79, 83)	(78, 75, 83)	(95, 92, 100)	(95, 92, 100)	(90, 87, 95)
s_3	(83, 80, 87)	(82, 79, 87)	(83, 80, 88)	(78, 75, 83)	(90, 87, 95)
s_4	(86, 83, 89)	(84, 81, 89)	(95, 92, 100)	(95, 92, 100)	(93, 90, 98)

Table 7: Percentage of commodity that remains usable after a disaster at each node (ρ_{jcs})

3.5 Results

3.5.1 Minimizing Objective 1

Results have shown that total of 10 RDCs (1 large and 9 small) should be built. The expected total cost is about 45.582 million, with the pre-disaster cost about 27.236 million and the expected post-disaster cost about 18.346 million. The locations and sizes of RDCs are as follows:

Location	Size	Location	Size	Location	Size
Gorgan	Small	Karaj	Small	Shahriar	–
Semnan	Large	Tehran	–	Qom	–
Sari	Small	Varamin	Small	Arak	Small
Rasht	Small	Robat Karim	–	Isfahan	Small
Qazvin	Small	Eslamshahr	–	Kashan	Small

Table 8: Locations and sizes of RDCs

The following figures show the pre-disaster allocation (green) and post-disaster flow (blue) from suppliers to RDCs under each scenario. Table 9 to Table 16 give all the results rounded off to the closest 10^3 units. Table 9 shows the pre-disaster amount of each commodity at each RDC (Q) and Table 10 shows the post-disaster amount of each commodity at each RDC under each scenario (X).

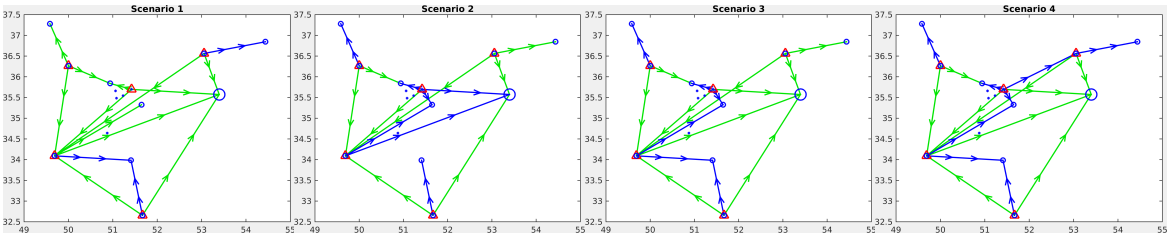

Figure 4: Pre-disaster allocation (green) and post-disaster flow (blue) from suppliers to RDCs

Supplier	Item	GO	SM	SA	RS	QZ	KR	VA	AR	IS	KS
Sari	Water	346	104	–	–	–	–	–	–	–	–
	Food	373	–	–	–	–	–	–	77	–	–
	Shelter	64	3	83	–	–	–	–	–	–	–
Qazvin	Water	–	–	–	–	450	–	–	–	–	–
	Food	–	–	–	–	–	–	–	450	–	–
	Shelter	–	–	–	83	66	0.2	–	–	–	–
Tehran	Water	–	–	–	–	–	510	–	–	–	–
	Food	–	–	–	–	–	–	–	510	–	–
	Shelter	–	106	–	–	–	64	–	–	–	–
Arak	Water	–	–	–	–	–	–	–	450	–	–
	Food	–	–	–	–	–	–	–	450	–	–
	Shelter	–	25	–	–	–	–	83	41	–	–
Isfahan	Water	–	–	–	–	–	–	–	–	450	–
	Food	–	–	–	–	–	–	–	21	428	–
	Shelter	–	62	–	–	–	–	–	–	4	83

Table 9: Amount of pre-disaster storage at each RDC in 10^3 units (Q)

Supplier	Item	Gorgan (s_1, s_2, s_3, s_4)	Semnan (s_1, s_2, s_3, s_4)	Sari (s_1, s_2, s_3, s_4)	Rasht (s_1, s_2, s_3, s_4)	Qazvin (s_1, s_2, s_3, s_4)
Sari	Water	(338, 0, 0, 0)	–	(0, 369, 405, 360)	–	–
	Food	(0, 0, 0, 47)	–	(271, 356, 392, 300)	–	–
	Shelter	–	–	(0, 131, 143, 128)	–	–
Qazvin	Water	–	–	–	(0, 383, 238, 0)	(0, 369, 154, 405)
	Food	–	–	–	(0, 0, 122, 0)	(0, 65, 256, 392)
	Shelter	–	–	–	(0, 71, 0, 87)	(0, 0, 138, 55)
Tehran	Water	–	–	(0, 0, 0, 182)	–	–
	Food	–	(0, 58, 0, 0)	–	–	–
	Shelter	–	–	–	–	–
Arak	Water	–	–	–	–	–
	Food	–	(0, 414, 0, 0)	–	–	–
	Shelter	–	–	–	–	–
Isfahan	Water	–	–	(0, 405, 0, 0)	–	–
	Food	–	–	(0, 392, 0, 0)	–	–
	Shelter	–	–	(0, 143, 0, 0)	–	–

Supplier	Item	Karaj	Varamin	Arak	Isfahan	Kashan
Sari	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	–	–	–
Qazvin	Water	(0, 134, 0, 0)	(0, 285, 0, 0)	–	–	–
	Food	(0, 345, 0, 0)	–	–	–	–
	Shelter	(0, 41, 0, 0)	(0, 106, 0, 0)	–	–	–
Tehran	Water	(0, 0, 95, 58)	(0, 428, 308, 198)	–	–	–
	Food	(0, 0, 388, 423)	–	–	–	–
	Shelter	(0, 0, 15, 27)	(0, 7, 128, 127)	(0, 143, 0, 0)	–	–
Arak	Water	–	(0, 0, 92, 407)	(0, 0, 251, 21)	–	(450, 0, 30, 0)
	Food	–	–	–	–	(0, 0, 360, 272)
	Shelter	–	–	(0, 0, 132, 123)	–	–
Isfahan	Water	–	–	–	(0, 244, 351, 197)	(450, 184, 0, 25)
	Food	–	–	–	(0, 0, 0, 414)	(0, 414, 338, 0)
	Shelter	–	–	–	(0, 99, 125, 71)	–

Table 10: Amount of post-disaster transfer of commodities from suppliers to RDCs in 10^3 units (X)

Table 11 shows the post-disaster transfer of commodities from RDCs to RDCs (Y) and Table 12 – 15 shows the post-disaster transfer of commodities from RDCs to AAs (Z) under each scenario. Below are the graphs for these transfers under each scenario:

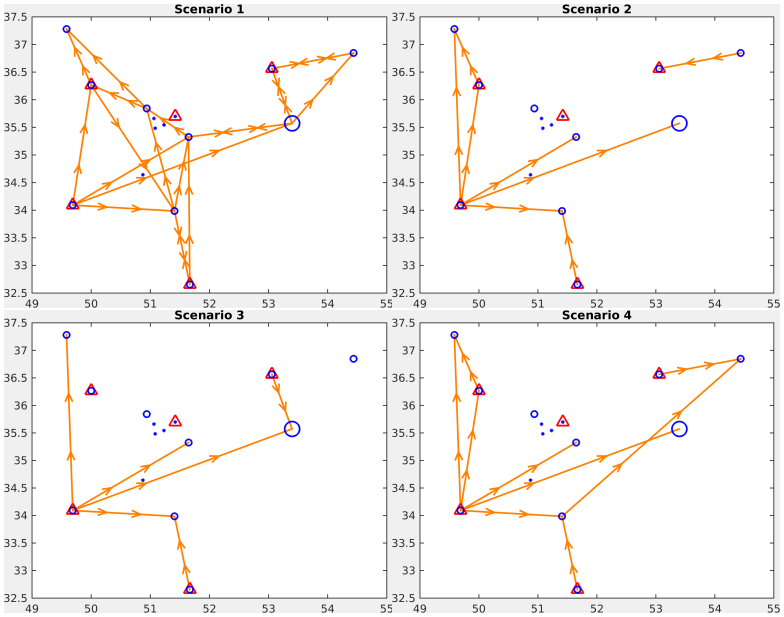

Figure 5: Post-disaster transfer of commodities from RDCs to RDCs (Y)

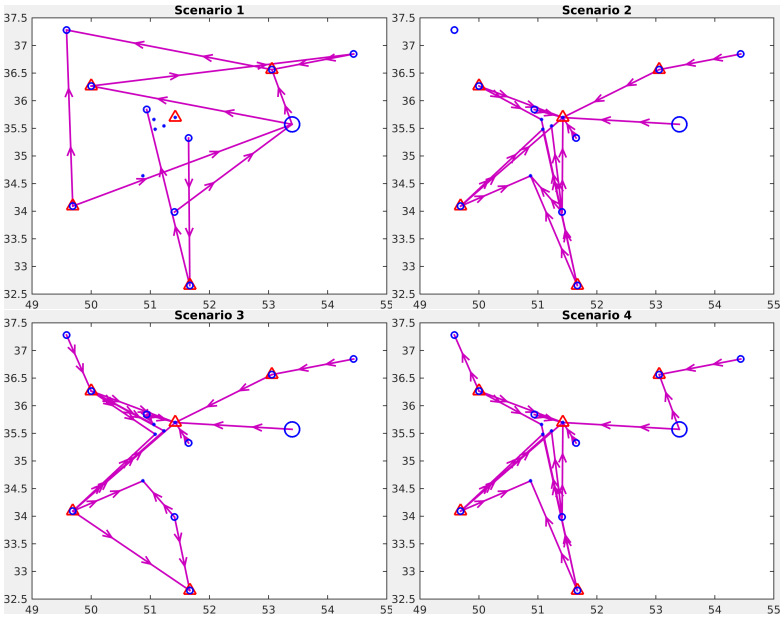

Figure 6: Post-disaster transfer of commodities from RDCs to AAs (Z)

RDC (j_1)	Item	Gorgan (s_1, s_2, s_3, s_4)	Semnan (s_1, s_2, s_3, s_4)	Sari (s_1, s_2, s_3, s_4)	Rasht (s_1, s_2, s_3, s_4)	Qazvin (s_1, s_2, s_3, s_4)
Gorgan	Water	(0, 0, 0, 956)	–	(39, 0, 0, 0)	–	–
	Food	–	–	(0, 72, 0, 0)	–	–
Semnan	Water	–	(0, 0, 536, 0)	(106, 0, 0, 0)	–	–
	Food	–	–	–	–	–
	Shelter	(51, 0, 0, 0)	(152, 856, 0, 718)	(126, 0, 0, 0)	–	–
Sari	Water	–	–	(729, 0, 0, 960)	–	–
	Food	(32, 0, 0, 0)	(239, 0, 18, 0)	–	–	–
	Shelter	(0, 0, 0, 40)	–	(0, 0, 982, 0)	–	–
Rasht	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	(518, 0, 0, 0)	–	–
Qazvin	Water	–	–	(379, 0, 0, 0)	(0, 45, 0, 365)	(0, 0, 1000, 0)
	Food	–	–	–	–	(0, 0, 0, 635)
	Shelter	–	–	–	–	(529, 0, 0, 0)
Karaj	Water	–	–	–	–	–
	Food	–	–	–	–	(68, 0, 0, 0)
	Shelter	–	–	–	(103, 0, 0, 0)	(25, 0, 0, 0)
Varamin	Water	–	(81, 0, 0, 0)	–	–	–
	Food	–	–	–	–	–
	Shelter	–	(20, 0, 0, 0)	–	–	–
Arak	Water	–	–	–	–	(248, 0, 0, 0)
	Food	–	(509, 144, 446, 282)	–	(0, 429, 116, 476)	(0, 170, 0, 45)
	Shelter	–	–	–	–	–
Isfahan	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	–	–	–
Kashan	Water	(0, 0, 0, 4)	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	–	–	–

RDC (j_1)	Item	Karaj (s_1, s_2, s_3, s_4)	Varamin (s_1, s_2, s_3, s_4)	Arak (s_1, s_2, s_3, s_4)	Isfahan (s_1, s_2, s_3, s_4)	Kashan (s_1, s_2, s_3, s_4)
Gorgan	Water	–	–	–	–	–
	Food	–	–	–	–	–
Semnan	Water	–	–	–	–	–
	Food	–	(169, 0, 0, 0)	–	–	–
	Shelter	–	–	–	–	–
Sari	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	–	–	–
Rasht	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	–	–	–
Qazvin	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	–	–	–	–	(91, 0, 0, 0)
Karaj	Water	–	–	–	–	–
	Food	–	–	–	–	–
	Shelter	(378, 0, 0, 1000)	–	–	–	–
Varamin	Water	(121, 0, 0, 0)	(201, 0, 0, 0)	–	–	–
	Food	(68, 0, 0, 0)	–	–	–	–
	Shelter	(64, 0, 0, 0)	(0, 0, 0, 809)	–	–	–
Arak	Water	–	(202, 0, 0, 0)	–	–	–
	Food	–	(0, 103, 397, 191)	–	–	(154, 0, 42, 6)
	Shelter	–	–	–	–	(41, 0, 0, 0)
Isfahan	Water	–	–	–	(503, 0, 0, 492)	–
	Food	–	(294, 0, 0, 0)	–	–	(134, 169, 151, 508)
	Shelter	–	–	–	(0, 831, 0, 0)	(4, 0, 0, 0)
Kashan	Water	(369, 0, 0, 0)	–	–	(497, 0, 0, 0)	(0, 677, 0, 0)
	Food	–	(134, 0, 0, 0)	–	(0, 0, 849, 0)	(0, 0, 807, 0)
	Shelter	–	–	–	–	(0, 0, 0, 487)

Table 11: Amount of post-disaster transfer of commodities from RDCs to RDCs in 10^3 units (Y)

RDC	Item	GO	SM	SA	RS	QZ	KR	TE	VA	RK	ES	SH	QM	AR	IS	KS
GO	Water	–	–	575	–	–	–	–	–	–	–	–	–	–	–	–
	Food	319	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	106	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SM	Water	–	–	4	–	68	–	–	–	–	–	–	–	–	–	–
	Food	–	–	575	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	11	–	–	23	–	–	–	–	–	–	–	–	–	–
SA	Water	–	–	–	145	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	193	–	–	–	–	–	–	–	–	–	–	–	–
RS	Water	–	–	–	379	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	175	–	–	–	–	–	–	–	–	–	–	–
QZ	Water	319	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	68	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
KR	Water	–	–	–	–	–	1000	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
VA	Water	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	529	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
AR	Water	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	476	–	524	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
IS	Water	–	–	–	–	–	947	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
KS	Water	–	34	–	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	221

Table 12: Amount of post-disaster transfer of commodities from RDCs to AAs in 10^3 units (Z) in s_1

RDC	Item	GO	SM	SA	RS	QZ	KR	TE	VA	RK	ES	SH	QM	AR	IS	KS
GO	Water	159	–	152	–	–	–	–	–	–	–	–	–	–	–	–
	Food	159	–	93	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	53	–	7951	–	–	–	–	–	–	–	–	–	–	–	–
SM	Water	–	69	–	–	–	–	23	–	–	–	–	–	–	–	–
	Food	–	69	–	–	–	–	547	–	–	–	–	–	–	–	–
	Shelter	–	23	–	–	–	–	159	–	–	–	–	–	–	–	–
SA	Water	–	–	369	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	428	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	166	–	–	–	37	–	–	–	–	–	–	–	–
RS	Water	–	–	–	428	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	429	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	143	–	–	–	–	–	–	–	–	–	–	–
QZ	Water	–	–	–	–	169	–	60	–	–	–	108	–	–	–	–
	Food	–	–	–	–	169	–	182	–	–	–	188	–	–	–	–
	Shelter	–	–	–	–	56	–	5	–	–	–	63	–	–	–	–
KR	Water	–	–	–	–	–	293	264	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	293	52	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	98	–	–	–	–	–	–	–	–	–
VA	Water	–	–	–	–	–	–	609	103	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	103	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	151	34	–	–	–	–	–	–	–
AR	Water	–	–	–	–	–	–	406	–	–	–	–	–	22	–	–
	Food	–	–	–	–	–	–	–	–	138	–	–	228	22	–	–
	Shelter	–	–	–	–	–	–	84	–	46	–	–	47	7	–	–
IS	Water	–	–	–	–	–	–	–	–	138	–	80	228	–	225	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	225	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	29	–	75	–
KS	Water	–	–	–	–	–	–	56	–	–	98	–	–	–	–	30
	Food	–	–	–	–	–	–	609	–	–	98	–	0	–	–	30
	Shelter	–	–	–	–	–	–	36	–	–	33	–	–	–	–	10

Table 13: Amount of post-disaster transfer of commodities from RDCs to AAs in 10^3 units (Z) in s_2

RDC	Item	GO	SM	SA	RS	QZ	KR	TE	VA	RK	ES	SH	QM	AR	IS	KS
GO	Water	96	–	229	–	–	–	–	–	–	–	–	–	–	–	–
	Food	96	–	243	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	32	–	32	–	–	–	–	–	–	–	–	–	–	–	–
SM	Water	–	29	–	–	–	–	70	–	–	–	–	–	–	–	–
	Food	–	29	–	–	–	–	435	–	–	–	–	–	–	–	–
	Shelter	–	10	–	–	–	–	186	–	–	–	–	–	–	–	–
SA	Water	–	–	60	–	–	–	345	–	–	–	–	–	–	–	–
	Food	–	–	46	–	–	–	328	–	–	–	–	–	–	–	–
	Shelter	–	–	64	–	–	–	157	–	–	–	–	–	–	–	–
RS	Water	–	–	–	238	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	238	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	79	1833	–	–	–	–	–	–	–	–	–	–
QZ	Water	–	–	–	–	158	–	210	–	–	–	177	–	–	–	–
	Food	–	–	–	–	158	–	–	–	–	–	98	–	–	–	–
	Shelter	–	–	–	–	51	–	32	–	32	25	59	–	–	–	–
KR	Water	–	–	–	–	–	222	316	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	222	86	–	–	–	79	–	–	–	–
	Shelter	–	–	–	–	–	74	–	–	–	–	–	–	–	–	–
VA	Water	–	–	–	–	–	–	320	81	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	316	81	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	176	27	–	–	–	–	–	–	–
AR	Water	–	–	–	–	–	–	187	–	100	76	–	187	75	–	–
	Food	–	–	–	–	–	–	–	–	100	–	–	32	75	–	–
	Shelter	–	–	–	–	–	–	–	–	0.9	–	–	62	25	81	–
IS	Water	–	–	–	–	–	–	–	–	–	–	–	–	–	702	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	170	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	128	–
KS	Water	–	–	–	–	–	–	–	–	–	–	–	155	–	–	30
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	706	30
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	69	10

Table 14: Amount of post-disaster transfer of commodities from RDCs to AAs in 10^3 units (Z) in s_3

RDC	Item	GO	SM	SA	RS	QZ	KR	TE	VA	RK	ES	SH	QM	AR	IS	KS
GO	Water	287	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Food	287	–	54	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	96	–	–	–	–	–	–	–	–	–	–	–	–	–	–
SM	Water	–	57	37	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	19	225	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	19	35	–	–	–	113	–	–	–	–	–	–	–	–
SA	Water	–	–	542	–	–	–	–	–	–	–	–	–	–	–	–
	Foo1d	–	–	300	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	158	–	–	–	–	–	–	–	–	–	–	–	–
RS	Water	–	–	–	365	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	476	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	159	–	–	–	–	–	–	–	–	–	–	–
QZ	Water	–	–	–	111	113	–	215	–	–	–	6	–	–	–	–
	Food	–	–	–	–	113	–	178	–	–	–	146	–	–	–	–
	Shelter	–	–	–	–	38	–	31	–	–	–	49	–	–	–	–
KR	Water	–	–	–	–	–	256	235	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	256	167	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	85	–	–	–	–	–	–	–	–	–
VA	Water	–	–	–	–	–	–	529	76	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	115	76	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	178	25	–	–	–	–	–	–	–
AR	Water	–	–	–	–	–	–	359	–	–	67	–	–	22	–	–
	Food	–	–	–	–	–	–	181	–	94	–	–	91	22	–	–
	Shelter	–	–	–	–	–	–	71	–	31	–	–	55	7	–	–
IS	Water	–	–	–	–	–	–	–	–	94	–	140	166	–	225	–
	Food	–	–	–	–	–	–	–	–	–	–	–	75	–	225	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	75	–
KS	Water	–	–	–	–	–	–	–	–	–	–	–	–	–	–	21
	Food	–	–	–	–	–	–	698	–	–	67	–	–	–	–	21
	Shelter	–	–	–	–	–	–	53	–	–	22	–	–	–	–	7

Table 15: Amount of post-disaster transfer of commodities from RDCs to AAs in 10^3 units (Z) in s_4

Finally, Figure 7 and Table 16 show the inventory and shortage at each AA under each scenario. Inventory (cyan) is positive and shortage (yellow) is negative.

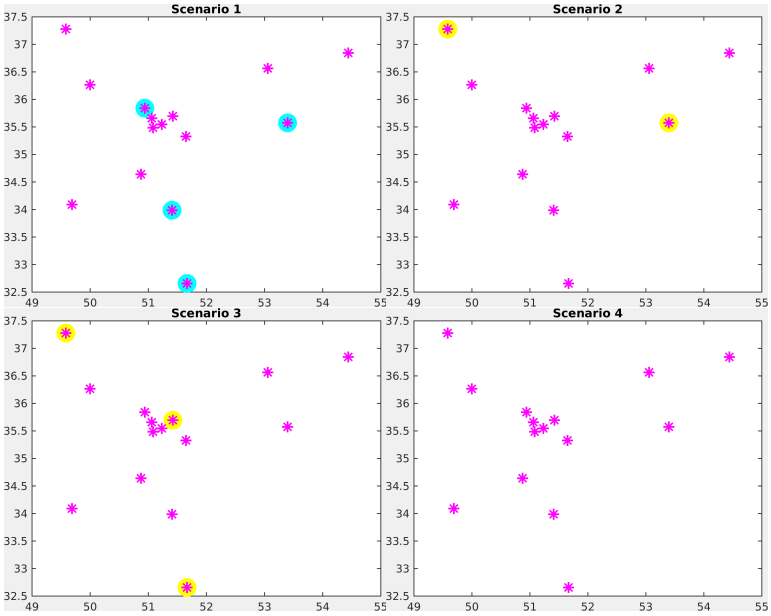

Figure 7: Post-disaster transfer of commodities from RDCs to RDCs (Y)

	Item	GO	SM	SA	RS	QZ	KR	TE	VA	RK	ES	SH	QM	AR	IS	KS
s_1	Water	–	–	–	–	–	1947	–	–	–	–	–	–	–	–	–
	Food	–	442	–	–	–	–	–	–	–	–	–	–	–	529	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	221
s_2	Water	–	–	–	-1	–	–	−0.6	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	−28	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
s_3	Water	–	–	–	–	–	–	−206	–	–		–	–	–	−288	–
	Food	–	–	–	–	–	–	−489	–	–	−76	–	–	–	−114	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	−52	–
s_4	Water	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Food	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
	Shelter	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Table 16: Inventory (I) and shortage (b) at each AA under each scenario in 10^3 units

In this report, we present only the results from the model with Objective 1 to demonstrate our solutiom methodoloy, However, the implementation with Objective 2 is similar and its solution will be examined in our future work.

3.6 System Performance

We demonstrate the solving time of two cases, which were randomly generated with a similar setting to our case study, on two computers, namely star1 and nanoheat, which contain 12 and 64 threads re-

spectively. Some statistics and benchmark of the performance of SYMPHONY on these two computers are as follows.

3.6.1 Small Case

The small case consists of 8 suppliers, 15 RDCs, 30 AAs, 3 sizes of RDCs, 20 scenarios and 3 types of commodities.

On star1, the total user time for this case study is approximately 2.39×10^3 seconds, that is about 11 times the total wallclock time (224 seconds).

On nanoheat, the total user time for this case study is approximately 1.07×10^4 seconds, that is about 59 times the total wallclock time (181 seconds).

3.6.2 Medium Case

The medium case consists of 10 suppliers, 20 RDCs, 80 AAs, 3 sizes of RDCs, 30 scenarios and 3 types of commodities.

On star1, the total user time for this case study is approximately 5.12×10^5 seconds, that is about 11 times the total wallclock time (4.66×10^4 seconds). Statistics for nanoheat is not available at this moment.

3.6.3 Large Case

The large case consists of 50 suppliers, 100 RDCs, 500 AAs, 3 sizes of RDCs, 10 scenarios and 3 types of commodities. At this point of research, we have not carried out a full run for this case. It is only known that neither lower nor upper bound of the problem could be found within 12 hours of run.

3.6.4 Observation

The total wallclock time is approximately inversely proportional to the number of threads. The total user time is approximately the same in each computer and parallel processing divides the user time required among the number of threads. Hence a higher number of threads favors a more efficient process in solving the linear program.

However, the relationship of the parameters size, number of threads and user time is not exactly known at this moment.

4 Future Work

Possible future work of this project includes speeding up the process for the program to solve the mixed-integer linear program and incorporating further variability in problem parameters to understand the robustness of solutoins. For higher efficiency, parallel processing is a solution because this allows the program to run multiple instances simultaneously. For robustness of solutions, uncertainty quantification and sensitivity analysis provide possibilities for more thoughtful and subtle plans to cope with the highly uncertain nature of disasters.

4.1 Parallel Processing

Results in Section 3.6 have suggested that parallel processing is helpful in increasing the efficiency of the solution procedure. Multiprocessing with a higher number of threads allows wallclock time of the solver to be reduced to a greater extent. Although the exact degree of improvement is unknown, how parallel processing could be implemented to this program is worth exploring. Once parallel processing is applied, a high number of similar cases could be prepared and solved at the same time, which would be extremely useful for sensitivity analysis and uncertainty quantification.

4.2 Uncertainty Quantification

Uncertainty quantification is the study of quantifying parameters in a mathematical model that are non-deterministic before an event occurs. In the case of disaster relief planning, the dynamic and uncertain nature of disasters leads to some variation in parameters that highly depend on scenarios and probabilities. Specifically, the parameters included, but not limited to, are as follows:

- Demands of each AA at each scenario
- Supplies available at each supplier and RDC
- Costs of items and transportation
- Discrepancy of collected data from actual numerical data

We rely on uncertainty quantification to simulate how a disaster and the corresponding parameters go. A possibility is to use PSUADE (Problem Solving environment for Uncertainty Analysis and Design Exploration), which is an interactive software designed for uncertainty quantification tasks.

4.3 Sensitivity Analysis

Sensitivity analysis is the study of determining how sensitive the output solution is to the input values. By the end of this project, we have obtained a preliminary understanding of sensitivity analysis by preparing 10 cases similar to that in Section 3.4 are created by varying the demand (D_{kcs}) and fraction of remained commodity (ρ_{ics} and ρ_{jcs}), using uniform distribution ranging from 50% to 150% of their original values.

Of course, for an in-depth study of sensitivity analysis, we need to prepare a much higer number of cases. Then, once parallel processing is ready, we could run multiple scenarios at a time, compare all the plans, and find the ultimate plan that fits all cases well.

5 Acknowledgments

This project is sponsored by the Chinese University of Hong Kong and Joint Institute for Computational Sciences. I would also like to thank my mentors Dr. Kwai Wong and Dr. Yong-Hong Kuo for their generous help and advice.

References

[1] Ali Bozorgi-Amiri, M. S. Jabalameli, and S. M. J. Mirzapour Al-e-Hashem. A multi-objective robust stochastic programming model for disaster relief logistics under uncertainty. *OR Spectrum*, 35:905–933, March 2013.

[2] Henry (Shang-Roh) Hsieh. Application of the PSUADE tool for sensitivity analysis of an engineering simulation. *Technical Report of the U. S. Department of Energy by Lawrence Livermore National Laboratory*, UCRL-TR-237205, 2007.

[3] T.K. Ralphs, M. Guzelsoy, and A. Mahajan. *SYMPHONY 5.6.9 User's Manual*, March 2015.

[4] Chian-Son Yu and H. L. Li. A robust optimization model for stochastic logistic problems. March 2000.