

OPTIMAL ROUTING IN A HIGH THREAT ENVIRONMENT: MODELS AND  
ALGORITHMS

---

A Thesis  
presented to  
the Faculty of the Graduate School  
at the University of Missouri-Columbia

---

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science

---

By  
JIHYUN JO  
Dr. Mustafa Sir, Thesis Supervisor

DEC 2010

Copyright by Jihyun Jo 2010

All Rights Reserved

The undersigned, appointed by the dean of the Graduate School, have examined the thesis entitled

**Optimal Routing in a High Threat Environment: Models and Algorithms**

presented by Jihyun Jo,

a candidate for the degree of master of science,

and hereby certify that, in their opinion, it is worthy of acceptance.

---

Professor Mustafa Sir

---

Professor Esra Sisikoglu

---

Professor Timothy Matisziw

## DEDICATION

To Dr. Sir M., Dr. Sunkyo Kim, faculty members,

To my friends,

And

To my family – mom, dad and my brother

## ACKNOWLEDGEMENTS

First of all, I would like to thank my thesis supervisor, Dr. Sir, M., who gave me lots of knowledge and helped me to finish this thesis. When I started to this research, I didn't have enough background knowledge and I didn't have any experiences of using MATLAB to solve the Markov Decision Problem. From his careful support, I can finish my work quite easily.

I also appreciate my friends, especially Pak, Yoon-Chee and Lin, Rung-Chuan. Yoon-Chee helped me to finish my coursework and to organize my thesis progress. Rung-Chuan gave me his MATLAB knowledge and shared the idea to finishing the coding.

And I also appreciate other friends who gave and shared their idea to me during my MS coursework. From their support, I can finish my entire work at the MIZZOU without any problems.

TABLE OF CONTENTS

ACKNOWLEDGEMENT ..... ii  
LIST OF ILLUSTRATIONS ..... iv  
LIST OF TABLES ..... vi  
ABSTRACT ..... vii

Chapter

1. INTRODUCTION .....1  
2. STRATEGIC MULTI-OBJECTIVE ROUTING IN A HIGH THREAT  
ENVIRONMENT USING INTEGER PROGRAMMING .....4  
3. DYNAMIC ROUTING IN A DYNAMICALLY CHANGING HOSTILE  
ENVIRONMENT: MODELS AND ALGORITHMS .....45  
4. CONCLUSION .....76  
  
REFERENCES .....78

APPENDIX

1. AMPL CODE .....81  
2. EXPERIMENTAL DESIGN (MULTI OBJECTIVE IP) .....89  
3. RISK CALCULATION TABLE .....93  
4. MULTI OBJECTIVE PROBLEMS RESULT TABLES .....99  
5. MATLAB CODE .....105  
6. LARGE SCALE PROBLEM RESULT OF MDP .....131

## LIST OF ILLUSTRATIONS

Figure	Page
1. Military statistics of Iraqi war .....	2
2. Various types of IEDs .....	3
3. Example of actual map for military operation .....	5
4. Simple example of risk zone .....	17
5. Simplified map for military operation.....	19
6. Route for minimum total distance problem.....	22
7. Route for min-max problem.....	22
8. Route for minimum risk problem.....	23
9. Weighted sum: $w_1 = 1, w_2 = 1, w_3 = 1$ .....	25
10. Weighted sum: $w_1 = 0.8, w_2 = 0.1, w_3 = 0.1$ .....	25
11. Weighted sum: $w_1 = 0.1, w_2 = 0.8, w_3 = 0.1$ .....	26
12. Weighted sum: $w_1 = 0.1, w_2 = 0.1, w_3 = 0.8$ .....	26
13. Route (total traveling distance goal: 1800) .....	28
14. Route (min-max flow goal: 165).....	29
15. Route (total risk goal: 470).....	29
16. Weighted sum goal programming: $w_1 = 1, w_2 = 1, w_3 = 1$ .....	32
17. Weighted sum goal programming: $w_1 = 0.8, w_2 = 0.1, w_3 = 0.1$ .....	32
18. Weighted sum goal programming: $w_1 = 0.1, w_2 = 0.8, w_3 = 0.1$ .....	33
19. Weighted sum goal programming: $w_1 = 0.1, w_2 = 0.1, w_3 = 0.8$ .....	33
20. Lexicographic route (priority: min risk > min distance > min-max flow) .....	35
21. Total risk changes (priority: min risk > min distance > min-max flow) .....	37
22. Total distance changes (priority: min risk > min distance > min-max flow) .....	37

23. Max flow changes (priority: min risk > min distance > min-max flow) .....	38
24. Lexicographic route (priority: min distance > min-max flow > min risk) .....	39
25. Total distance changes (priority: min distance > min-max flow > min risk) .....	40
26. Max flow changes (priority: min distance > min-max flow > min risk).....	40
27. Total risk changes (priority: min distance > min-max flow > min risk) .....	40
28. Max flow changes (priority: min-max flow > min risk > min distance).....	42
29. Total risk changes (priority: min-max flow > min risk > min distance) .....	42
30. Total distance changes (priority: min-max flow > min risk > min distance) .....	42
31. Lexicographic route (priority: min-max flow > min risk > min distance).....	43
32. Simple map for MDPs.....	46
33. State information at the beginning of time t.....	51
34. Possible action sets during time t .....	52
35. Enemy's individual movement.....	52
36. Enemies' location changing and their transition probability values .....	57
37. DVI approach .....	61
38. Modified DVI approach .....	65
39. Map for experiment (initial state).....	66
40. Value iteration result .....	67
41. Value iteration results of single enemy cases.....	69
42. DVI result .....	70
43. Modified DVI result .....	71
44. Total computation time result .....	73
45. Average number of encounters.....	73
46. Average number of step sizes.....	74
47. Average total costs .....	74



## LIST OF TABLES

Table	Page
1. Distance table .....	20
2. Risk table.....	21
3. Optimization result of single objective .....	22
4. Optimization result of weighted sum .....	25
5. Result of single objective goal programming .....	29
6. Result of multi objective goal programming.....	32
7. Results of optimization process (min risk > min distance > min-max) .....	37
8. Results of optimization process (min distance > min-max > min risk) .....	39
9. Results of optimization process (min-max > min risk > min distance ) .....	41
10. MDP: Brief results of experiment .....	72

# OPTIMAL ROUTING IN A HIGH THREAT ENVIRONMENT: MODELS AND ALGORITHMS

Jihyun Jo

Dr. Mustafa Sir, Thesis Supervisor

## ABSTRACT

In this thesis, we develop routing models and algorithms in a high threat environment. Although routing problems have been studied extensively in many other contexts, they are usually designed for only one objective: minimizing total cost (distance). However, many other factors, such as risk and path diversification, must be taken into account while routing in a high threat environment.

In this thesis, we consider two approaches to solve the routing problem in a high threat environment. In the first approach, we use a multi-objective integer programming to find best routes for troops from bases to target area given a transportation network. Objective functions we consider include minimizing total distance, total risk, and maximum flow on a given transportation arc. The main contribution of the first approach is quantification of risk given static locations of potential improvised explosive device attacks.

In the second approach, we develop a Markov decision model to dynamically route a troop in a dynamically changing hostile environment. We solve it optimally for a small problem instance using value iteration algorithm. For larger instances, we introduce a novel approximation scheme for the underlying dynamic program. Numerical experiments show that our approximation gives near optimal routing policies efficiently.

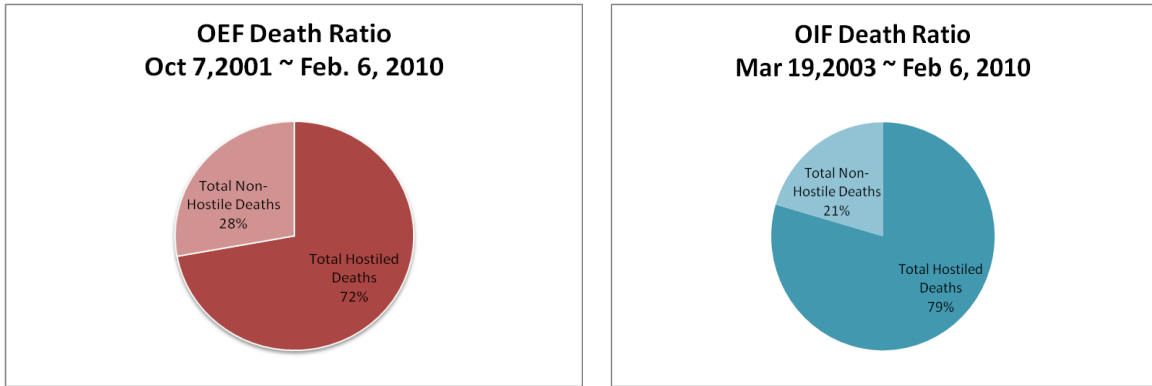
# CHAPTER 1. INTRODUCTION

---

According to the military statistics related to OIF (operation Iraqi freedom) and OEF (operation enduring freedom) from Congressional Research Service (2010), total of 3469 (from March 19, 2003 to February 6, 2010) and 702 (from October 7, 2001 to February 6, 2010) soldiers were killed due to the hostile activity. These numbers are more than 70% of total deaths related to those wars. Moreover, currently, total number of soldiers who were wounded in action is 4,939 (from March 19, 2003 to February 6, 2010) and 31,651 (from October 7, 2001 to February 6, 2010). The important fact that we have to consider is that many of these cases resulted from improvised explosive device (IED) attacks. Based on icasualties.org database, more than 40% of hostile deaths came from IED attack after 2005 and its ratio is more than 60% in 2005 and 2006. (see Figure 1)

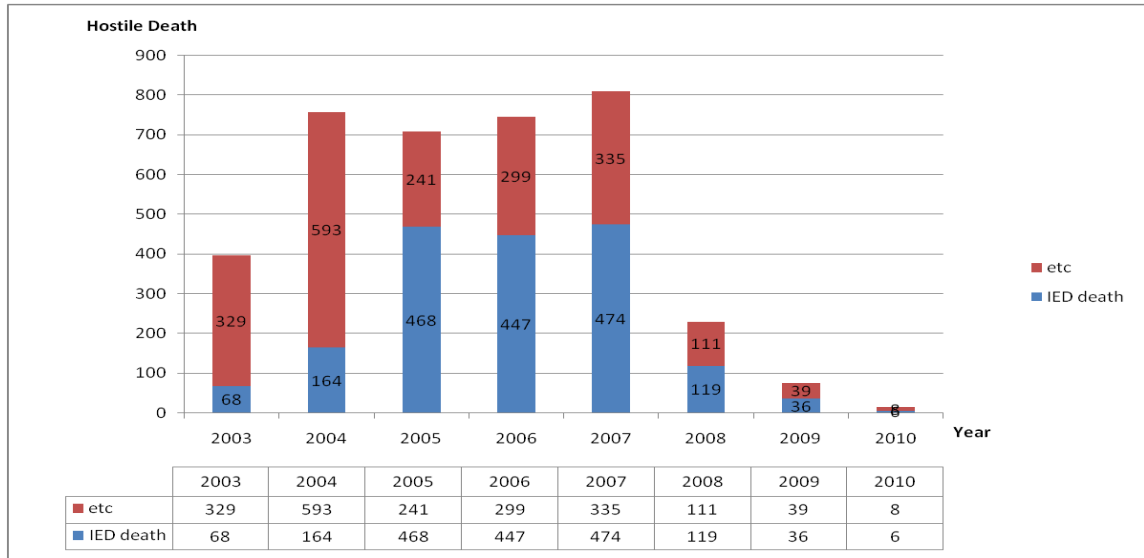
One example of IEDs is home-made bomb. There are various types of IEDs with different levels of destructive power. Since there is the asymmetry of military forces and

capabilities, insurgents use the IEDs to match these differences because they are cheap and simple to make and give effective results. Furthermore, since the development of IED advances continuously, military needs to deal with this kind of attack when making action plans in the battlefield. (see Figure 2)



(a) OEF death ratio

(b) OIF death ratio



(C) Military death trend (2003 ~ 2010)

Figure 1. Military statistics of Iraqi war



Figure 2. Various types of IEDs

In this thesis, we consider two related problems;

1. For a given transportation network, what are the best routes for troops from bases to target area considering risk of encountering hostile attack?
2. How do we dynamically route a troop in a dynamically changing hostile environment?

The thesis is organized as follows: In chapter 2, we will describe the strategic multi-objective routing using Integer Programming. In chapter 3, we will model the dynamic programming framework and develop several efficient approximation strategies (heuristics) to solve it.

# **CHAPTER 2. STRATEGIC MULTI-OBJECTIVE ROUTING IN A HIGH THREAT ENVIRONMENT USING INTEGER PROGRAMMING**

---

## **2.1 Problem Description**

There are many objectives for military routing problem but current routing technologies are mainly considering the minimizing transportation cost. That is; current technologies do not focus on properly assess the tradeoffs between multi-objectives such as minimizing risk while minimizing traveling distance. Moreover, the solution of the

single objective problem (i.e. minimum cost routing problem) is relatively predictable, so this approach may lead to strategic vulnerabilities in the battlefield.

In this chapter, we will discuss the multi-objective routing problem on the battlefield where the strategic objectives include minimizing maximum path flow, minimizing total distance, and minimizing total risk. In other words, we develop a model for a commander who wants to move his troops from military bases to mission areas as quickly as possible while minimizing risk of an attack. (see Figure 3)

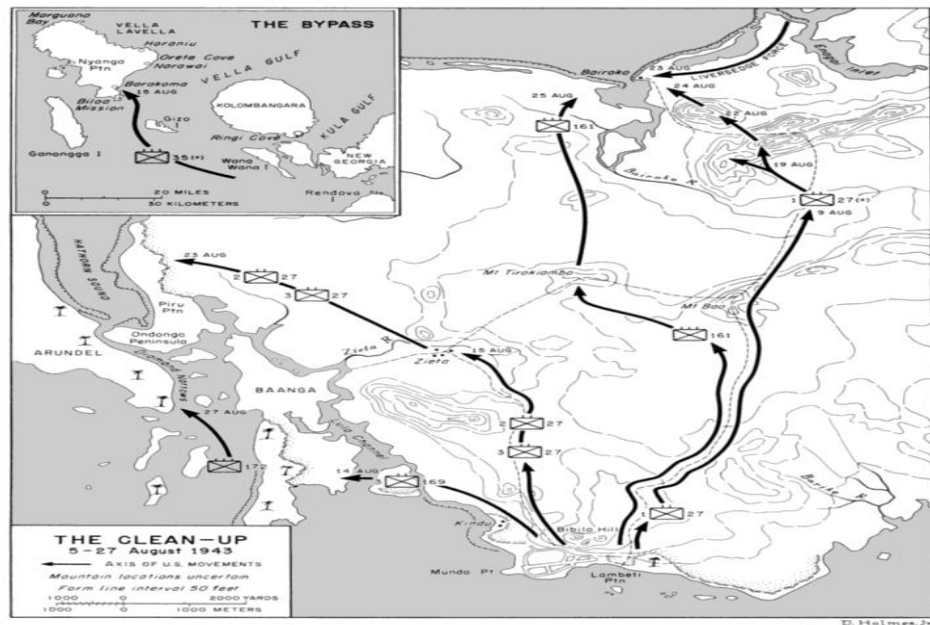


Figure 3. Example of actual map for military operation

Our model is based on the traditional transportation network problem, where the network consists of a given set of origins (e.g. military bases), intermediate transportation nodes, and destinations (e.g. battlefields or mission areas). Each military base has multiple troops and each battlefield or mission area requires a certain minimum number of troops to execute the missions successfully.

## 2.2 Literature review

One challenge when determining strategic routes for the troops is to quantitatively measure risk associated with a set of routes. Several researches attempted to model risk in transportation problem as follows. Murray-Tuite and Mahmassani (2004), for example, introduced the vulnerability indices which are obtained from a function of alternate paths, excess capacity and travel time. From these indices, they generated disruption indices which are network level importance measure of a link. Murray-Tuite (2008) defined risk as a probability with some measure of consequences. In her work, calculation of risk incorporates attack frequency, likelihood of a successful attack, Moteff's threat (which is based on intent, resources, capability, and history), and Haimes's threat (which is malicious entity) concepts.

In the Operation Research literature, risk has been incorporated into many models. One example is routing in a weather system. Vedat, Amit and Batta (2007) assumed that risk and travel time are dependent due to weather system. Risk concept they used includes expected consequences, impact, and incident probability. They also assumed that the distance from the center of a weather system, which is assumed to have a circular shape, affects the risk. Zografos and Androutsopoulos (2004) defined that risk is a combination of accident probability and population within a certain distance from an arc segment. In military applications, Carlyle, Royset, and Wood (2007) used a set of "threat circles" which are centered on the ground threats' locations to describe the risk and assumed that the risk values within the circle are determined by distance from center.



They also assumed that activity of each threatening element is independent and local, which allowed them to define simple functions for risk parameter calculation.

Instead of using a truly multi-objective approach, Zografos and Androutsopoulos (2004) used a weighted-sum approach when modeling the routing problem in hazardous materials distribution problems. They considered two objectives: 1) minimizing total travel time and 2) minimizing risk associated with a route. To handle multiple objectives, they also use the traditional way of weighted sum approach. Stepanov and Smith (2007) also use a weighted sum to solve their bi-objective problem where the objectives include minimizing total distance and minimizing total clearance time. In their paper, they used a modified version of the weighted sum approach because each objective has different measures. To deal with this, they divided each objective by the corresponding optimal value resulting from the single objective model. These are then combined as a weighted sum.

In the material handling problem area, Wadhwa and Ravindran used the various approached to solve their tri-objective problem; which includes 1) minimizing total purchasing cost, 2) minimizing lead time, and 3) minimizing reject rate. They used weighted-sum, goal programming, and compromise programming approach. They also compared the each method with some criterions, so we picked the suitable solution approaches, weighted sum and goal programming, for our model.

In the transportation research area, Sun, Ritchie, Tsai and Jayakrishnan(1999) used the lexicographic optimization for vehicle reidentification problem. Their formulation has 5 levels: first level for searching for vehicle pair (upstream vehicle),

second and third level for selection of vehicles, fourth level for minimizing discrepancies between upstream and downstream in a vehicle waveform pair using utility function, and 5<sup>th</sup> level for finding appropriate distance measure. We focus on the first three lexicographic goal programming approach for our model.

## 2.3 Formulation of Strategic Multi-Objective Routing Problem

### 2.3.1 Model

#### Sets and parameters

- $\mathcal{N}$ : set of nodes
- $\mathcal{B}$ : set of bases
- $\mathcal{J}$ : set of intermediary transportation nodes
- $\mathcal{T}$ : set of targets
- $\mathcal{N} \equiv \mathcal{B} \cup \mathcal{J} \cup \mathcal{T}$
- $\mathcal{M}$ : set of elements
- $d_{ij}$ : distance from node  $i \in \mathcal{N}$  to node  $j \in \mathcal{N}$ ,  $i \neq j$
- $a_{bt}$ : supply, number of  $t$  type element ( $t \in \mathcal{M}$ ) available at base  $b \in \mathcal{B}$
- $\tau_{st}$ : demand, required number of  $t$  type element ( $t \in \mathcal{M}$ ) for target  $s \in \mathcal{T}$
- $r_{ij}$ : risk measure associated with arc from node  $i \in \mathcal{N}$  to node  $j \in \mathcal{N}$ ,  $i \neq j$

#### Decision Variables

- $x_{ijt}$ : number of  $t$  type of elements ( $t \in \mathcal{M}$ ) node  $i \in \mathcal{N}$  to node  $j \in \mathcal{N}$ ,  $i \neq j$

As mentioned above, this model is based on the traditional transportation problem. The decision variables are defined as the number of elements moving from

node  $i$  to node  $j$ . We assume that the each element cannot partially move from node  $i$  to node  $j$ . Therefore, the problem is modeled as a pure integer programming problem.

**Objective functions:**

- Minimizing total distance traveled by all troops:

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} d_{ij} x_{ijt}$$

This is a typical objective in the classical transportation problem [10].

- Minimizing maximum flow (weighted with respect to arc length) on an arc:

$$\min \sum_{t \in \mathcal{M}} M(t)$$

with additional constraints

$$\sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} d_{ij} x_{ijt} \leq M(t)$$

$$M(t) \geq 0$$

- Minimizing risk:

$$\min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} r_{ij} x_{ijt}$$

This objective is a function of risk measures  $r_{ij}$  associated with each arc. In section 2.4, we discuss how risk measures are calculated.

## Constraints:

Constraints are similar to the traditional transportation network problem. Constraints include resource availability constraints, resource requirement constraints, and node balancing constraints. Additional constraints include the definition constraints for  $M(t)$  that represent maximum flow on an arc.

## Mathematical Model

$$\left\{ \begin{array}{l} \min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} d_{ij} x_{ijt} \\ \min \sum_{t \in \mathcal{M}} M(t) \\ \min \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} r_{ij} x_{ijt} \end{array} \right\}$$

Subject to

$$\sum_{b \in B, j \in I} x_{bjt} \leq a_{bt} \text{ (Elements availability at military base } b \text{)}$$

$$\sum_{i, k \in I} x_{ikt} - \sum_{k, j \in I} x_{kjt} = 0 \text{ (Node balance within intermediary nodes)}$$

$$\sum_{j \in I, s \in T} x_{jst} \geq \tau_{st} \text{ (Elements requirement at battlefield } T \text{)}$$

$$\sum_{i, j \in \mathcal{N}} d_{ij} x_{ijt} \leq M(t) \text{ (Definition of } M \text{)}$$

$$x_{ijt} \geq 0, \text{ integer}$$

### 2.3.2 Solution approaches:

There are various methods to solve the multi-objective problem. In this paper, we use three different multi-objective approaches: weighted sum, goal programming and lexicographic ordering.

#### Weighted Sum Approach

Weighted sum [2], [14] is the most frequently used approach to solve multi-objective optimization problem because of its simplicity. In this approach, multiple objectives are combined into a single objective as a weighted sum:

$$\min w_1 \left( \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} d_{ij} x_{ijt} \right) + w_2 \left( \sum_{t \in \mathcal{M}} M(t) \right) + w_3 \left( \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} r_{ij} x_{ijt} \right),$$

where  $\sum_{i=1}^3 w_i = 1$ , and  $w_i \geq 0, i = 1, 2, 3$ .

The challenge when using a weighted sum approach is to find proper weights for each objective. Since the scale of each objective is different, we use modified weights:

$$\min \frac{w_1}{\bar{D}} \left( \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} d_{ij} x_{ijt} \right) + \frac{w_2}{\bar{M}} \left( \sum_{t \in \mathcal{M}} M(t) \right) + \frac{w_3}{\bar{R}} \left( \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}, i \neq j} r_{ij} x_{ijt} \right),$$

where  $\sum_{i=1}^3 w_i = 1, w_i \geq 0, i = 1, 2, 3$  and  $\bar{D}, \bar{R}, \bar{M}$  is optimal values for single objective problems of minimizing total distance, minimizing maximum flow, and minimizing total risk, respectively.

## Goal Programming

Based on [8], [16], the objective of the goal programming approach is to minimize deviations from pre-specified targets for each objective. The goal programming approach usually involves three steps:

- a. Decision maker sets a goal for each objective and defines goal constraints.
- b. Decision maker determines a preference for each objective.
- c. Find an optimal solution set using the modified model based on the previous two steps.

The modified model is given as follows:

$$\min p_1 d_1^+ + p_2 d_2^+ + p_3 d_3^+$$

Subject to

$$\sum_{b \in B, j \in I} x_{bjt} \leq a_{bt} \text{ (Elements availability at base b)}$$

$$\sum_{i, k \in I} x_{ikt} - \sum_{k, j \in I} x_{kjt} = 0 \text{ (Node balance)}$$

$$\sum_{j \in I, s \in T} x_{jst} \geq \tau_{st} \text{ (Elements requirement at target area)}$$

$$\sum_{i, j \in N} d_{ij} x_{ijt} \leq M(t) \text{ (Definition of M)}$$

$$\sum_{i, j \in N} d_{ij} x_{ij} + d_1^- - d_1^+ = \text{(distance goal)}$$

$$\sum_{i, j \in N} R_{ij} x_{ij} + d_2^- - d_2^+ = \text{(risk goal)}$$

$$M + d_3^- - d_3^+ = \text{(min - max flow goal)}$$

$$x_{ijt} \geq 0, \text{ integer}$$

$$d_i^-, d_i^+ \geq 0, i = 1,2,3$$

In the new objective function,  $p_i, i = 1,2,3$ , are the preferences, which allow the decision maker to rank order three objectives according to his/her priorities. New variables  $d_i^-$  and  $d_i^+$  are the negative and positive deviations from goal  $i, i = 1,2,3$ , respectively. In the modified objective, we minimize a weighted-sum of the positive deviations since all of the original objectives are minimization.

### **Lexicographic Goal Programming**

Based on [7], [17], lexicographic ordering method solves multi-objective problem as single objective problem sequentially in order of priorities. There are several advantages of lexicographic ordering approach over weighted sum approach: one is objectives in different units can be split by different priority levels. This means we can avoid the assigning weights to the objectives that have different unit scales. Another advantage is that a separate problem is solved for each objective sequentially. This way, sensitivity of solutions to a specific level of objectives can be found.

In the lexicographic ordering:

$$\min f_i(x)$$

Subject to

$$f_j(x) \leq f_j(x_j^*)$$



where  $i = 1, 2, \dots, N, j = 1, 2, \dots, i - 1$ , and  $x_j^*$  is solution set of  $j$ th criteria.

However, there are some drawbacks of this method: order determination problem and robustness of solution set. This leads that less important objectives are sometimes ignored. To overcome these problems, we use the lexicographic goal programming which is similar to the lexicographic ordering.

In the lexicographic goal programming:

$$\min f_i(x)$$

Subject to

$$f_j(x) \leq T_j,$$

where  $i = 1, 2, \dots, N, j = 1, 2, \dots, i - 1$ , and  $T_j = (1 + \alpha_j) * f_j(x_j^*)$  is a  $j$ th criteria target value.

## 2.4 Calculation of risk measure

It is challenge to define what risk means and quantitatively calculate a risk factor.

In our model, we consider risk zones having circular shapes, which alternately determines risk factors associated with each arc on a transportation network. This method has been used by other authors [1, 2] due to its simplicity and close proximity to real-world situations. We assume that a possible IED attack point forms the center of risk zones, and that the center is the riskiest point within a risk zone. The further a point from the center, the lesser is the risk value associated with it. The radius of risk zone is related to the affected area when an actual attack occurs and this value is predetermined based on damage caused by previous attacks.

Secondly, we consider the topography of the area. Although we do not consider the altitude of the ground, configuration of ground can significantly affect the result of IED attack. For example, the explosion in an open area (i.e. public square, playground, or etc) has different impact that the one in a closed area (i.e. woodland, forest, or etc).

Last factor we consider is the impact of an IED attack. As we discussed in Chapter 1, there are different types of IEDs and having different destruction power. In our model, we assume that we know all the possible IEDs that enemies can use and historical data of attack frequencies and tendency. This information can be used to calculate impact of a potential IED attack. For simplicity, we will use simple expected

value of IED attack result using IEDs power and probability of IED that can be used in a certain risk zone.

Figure 4 illustrates a transportation arc going through a risk zone associated with a potential IED attack point at location  $c$  having radius  $r$ .

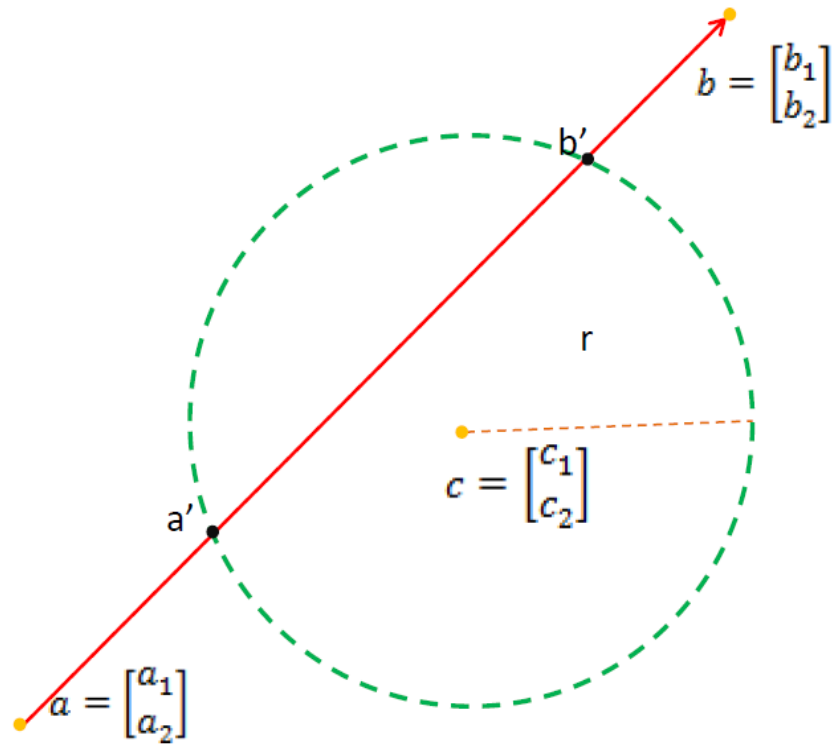


Figure 4. Simple example of risk zone

The risk associated with this arc is calculated as follows:

$$R = \frac{\alpha \cdot \beta}{h},$$

where let  $\bar{x}$  be such that

$$\bar{x} = \frac{\|a' - a\|}{\|a - b\|}$$

and  $\bar{x}$  be such that

$$\bar{x} = \frac{\|b' - a\|}{\|a - b\|}$$

- $h = \int_{\bar{x}} \|(b - a)x + a - c\| dx$ .  $h$  can be thought of as a measure of how close the car passes by the IED attack point.
- $\alpha$ : topography of the risk zone (e.g.  $0.8 \leq \alpha \leq 1.2$ )
- $\beta$ : impact of IED attack at the risk zone.

Impact is calculated as

$$\beta = \sum_{i \in S} p_i d_i$$

where  $S$  is the set of possible IEDs used in an attack, and  $p_i$  is the probability of using IED  $i \in S$  and  $d_i$  is destructive power of IED  $i \in S$ , respectively.

If an arc passes through multiple risk zones, the same calculations are made for each risk zone and the risk factors are then added together which results in an overall risk factor for that arc. For easy interpretation risk factors are normalized across all arcs.

## 2.5 Numerical experiment

### Experimental design

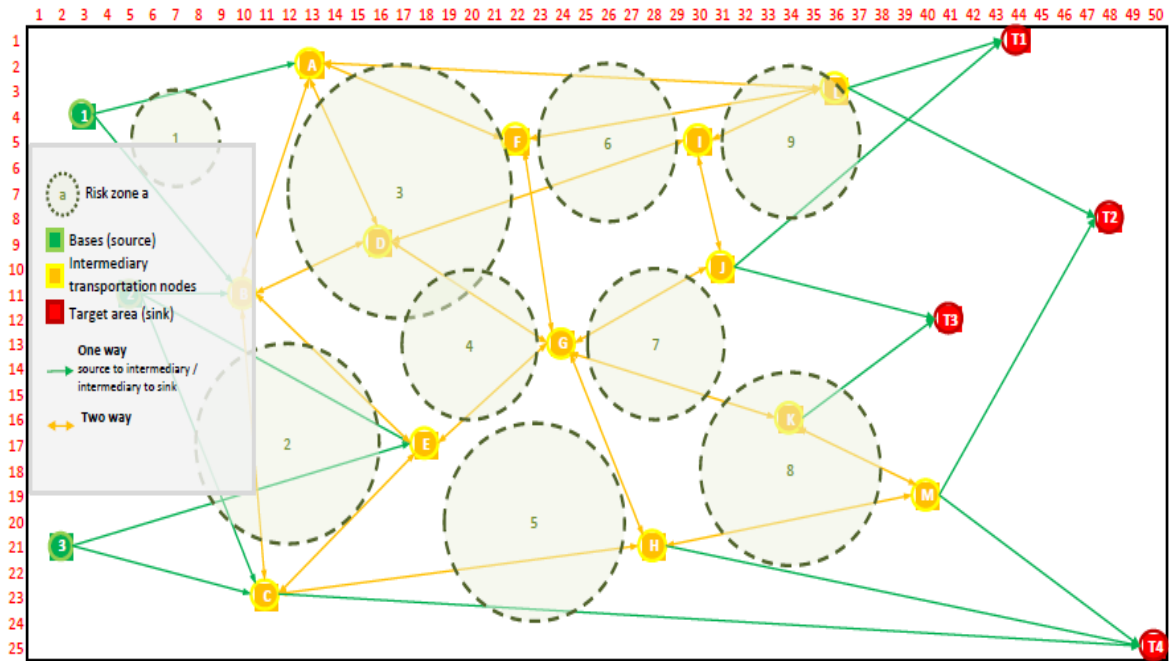


Figure 5. Simplified map

- Assumptions
  - 3 types of elements that we need to supply from sources to destinations
  - Sources and destinations: 3 bases and 4 target areas
  - Base capacity

Sources	tk	bb	ms
1	25	10	5
2	20	25	30
3	5	15	10

- Operation requirement

Targets	tk	bb	ms
T1	15	10	10
T2	10	15	10
T3	5	10	15
T4	10	5	5

- Arcs: 2 types of arcs
  - One way arcs
    - Between sources and intermediary nodes
    - Between intermediary nodes and targets
  - Two way arcs: between intermediary nodes
- Total 37 arcs:
  - 16 one way arcs
  - 21 two way arcs: 42 possible paths
- Risk Zones: 23 arcs (39 paths) passing through total 9 risk zones
- Parameter tables: distance and risk information b/w nodes

Distance Table

	1	2	3	A	B	C	D	E	F	G	H	I	J	K	L	M
A	10.2	1000.0	1000.0	1000.0	9.5	1000.0	7.6	1000.0	9.5	1000.0	1000.0	1000.0	1000.0	1000.0	23.0	1000.0
B	9.9	5.0	1000.0	9.5	1000.0	12.0	6.3	10.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0
C	1000.0	13.4	9.2	1000.0	12.0	1000.0	1000.0	9.2	1000.0	1000.0	17.1	1000.0	1000.0	1000.0	1000.0	1000.0
D	1000.0	1000.0	1000.0	7.6	6.3	1000.0	1000.0	1000.0	8.9	1000.0	14.6	1000.0	1000.0	1000.0	1000.0	1000.0
E	1000.0	14.3	16.5	1000.0	10.0	9.2	1000.0	1000.0	7.2	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0
F	1000.0	1000.0	1000.0	9.5	1000.0	1000.0	1000.0	1000.0	8.2	1000.0	1000.0	1000.0	1000.0	1000.0	14.1	1000.0
G	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	8.9	7.2	8.2	1000.0	8.9	1000.0	7.6	10.4	1000.0	1000.0
H	1000.0	1000.0	1000.0	1000.0	1000.0	17.1	1000.0	1000.0	8.9	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	12.2
I	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	14.6	1000.0	1000.0	1000.0	1000.0	1000.0	5.1	1000.0	6.3	1000.0
J	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	7.6	1000.0	5.1	1000.0	1000.0	1000.0	1000.0	1000.0
K	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	10.4	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	6.7
L	1000.0	1000.0	1000.0	23.0	1000.0	1000.0	1000.0	1000.0	14.1	1000.0	1000.0	6.3	1000.0	1000.0	1000.0	1000.0
M	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	12.2	1000.0	6.7	1000.0	1000.0	1000.0
T1	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	15.8	1000.0	8.2	1000.0
T2	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	13.0	13.6
T3	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	1000.0	10.2	8.1	1000.0	1000.0
T4	1000.0	1000.0	1000.0	1000.0	1000.0	39.1	1000.0	1000.0	1000.0	1000.0	22.4	1000.0	1000.0	1000.0	1000.0	11.7

Table 1. distance table

Risk Table (From: Column / To: Row)

	1	2	3	A	B	C	D	E	F	G	H	I	J	K	L	M
A	1.31	1000.00	1000.00	1000.00	0.00	1000.00	3.62	1000.00	3.61	1000.00	1000.00	1000.00	1000.00	1000.00	12.71	1000.00
B	0.00	0.00	1000.00	0.00	1000.00	2.90	0.00	2.94	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
C	1000.00	2.94	0.00	1000.00	2.90	1000.00	1000.00	2.97	1000.00	1000.00	5.07	1000.00	1000.00	1000.00	1000.00	1000.00
D	1000.00	1000.00	1000.00	3.61	0.00	1000.00	1000.00	1000.00	1000.00	4.04	1000.00	3.93	1000.00	1000.00	1000.00	1000.00
E	1000.00	2.88	2.97	1000.00	2.94	2.97	1000.00	1000.00	1000.00	3.39	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
F	1000.00	1000.00	1000.00	3.61	1000.00	1000.00	1000.00	1000.00	1000.00	0.00	1000.00	1000.00	1000.00	1000.00	9.00	1000.00
G	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	4.04	3.39	0.00	1000.00	0.00	1000.00	0.99	2.47	1000.00	1000.00
H	1000.00	1000.00	1000.00	1000.00	1000.00	5.07	1000.00	1000.00	1000.00	0.00	1000.00	1000.00	1000.00	1000.00	1000.00	6.62
I	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	3.93	1000.00	1000.00	1000.00	1000.00	1000.00	0.00	1000.00	6.23	1000.00
J	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	0.99	1000.00	0.00	1000.00	1000.00	1000.00	1000.00
K	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	2.47	1000.00	1000.00	1000.00	1000.00	1000.00	6.63
L	1000.00	1000.00	1000.00	12.71	1000.00	1000.00	1000.00	1000.00	9.00	1000.00	1000.00	6.23	1000.00	1000.00	1000.00	1000.00
M	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	6.62	1000.00	1000.00	6.63	1000.00	1000.00
T1	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	6.29	1000.00	0.00	1000.00
T2	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	0.00	0.00
T3	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	0.00	1.34	1000.00	1000.00
T4	1000.00	1000.00	1000.00	1000.00	1000.00	5.15	1000.00	1000.00	1000.00	1000.00	0.00	1000.00	1000.00	1000.00	1000.00	0.00

Table 2. risk table

- Computation

To solve the problem, we made AMPL codes and used the CPLEX option to handle the integer programming. The computing system was Intel® Xeon® CPU (E5330 @ 2.40GHz (2 processors)) and 32.0GB ram with 64-bit windows 7. Most of the problem is solved less than 5 sec.

## Weighted sum approach

We first solve the transportation problem using single objective. Then, multiple objectives are combined using a weighted sum approach.

- Optimization using single objective: minimizing total distance

When we only consider the minimizing total traveling distance, objective value is 5135.5 and assigned elements on the arcs are shown in Figure 6. This solution results on a maximum flow of 690 (between node A and node L) and total risk of 1120.85.

- Optimization using single objective: minimizing maximum flow

When we only consider the minimizing maximum flow, objective value is 437.1 (between node C and node H) and assigned elements on the arcs are shown in Figure 7. This solution results on a total distance of 5676.2, and total risk of 1386.67.

- Optimization using single objective: minimizing risk

When we only consider the minimizing risk, objective value is 915.9 and assigned elements on the arcs are shown in Figure 8. This solution results on a total distance of 5550, and a maximum flow of 1460 (between node D and node I).

objective	min total distance	min maximum flow	min total risk
total distance	5135.5	5676.2	5550
maximum flow	690	437.1	1460
total risk	1120.85	1386.67	915.9

Table 3. Optimization result of single objective



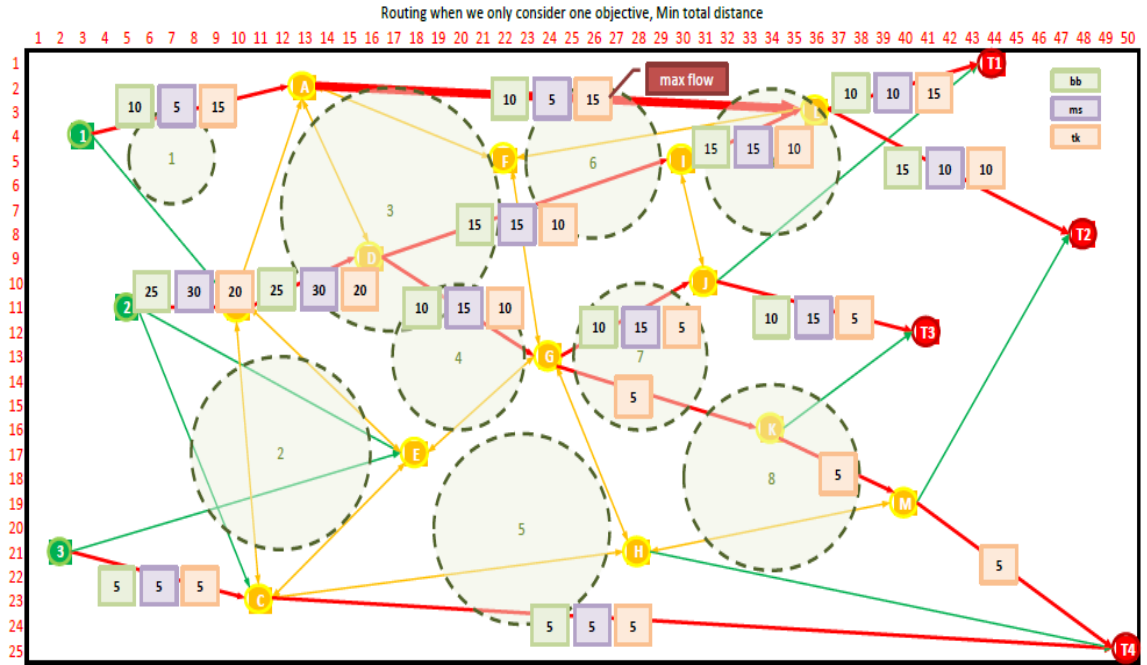


Figure 6. Route for minimum total distance problem

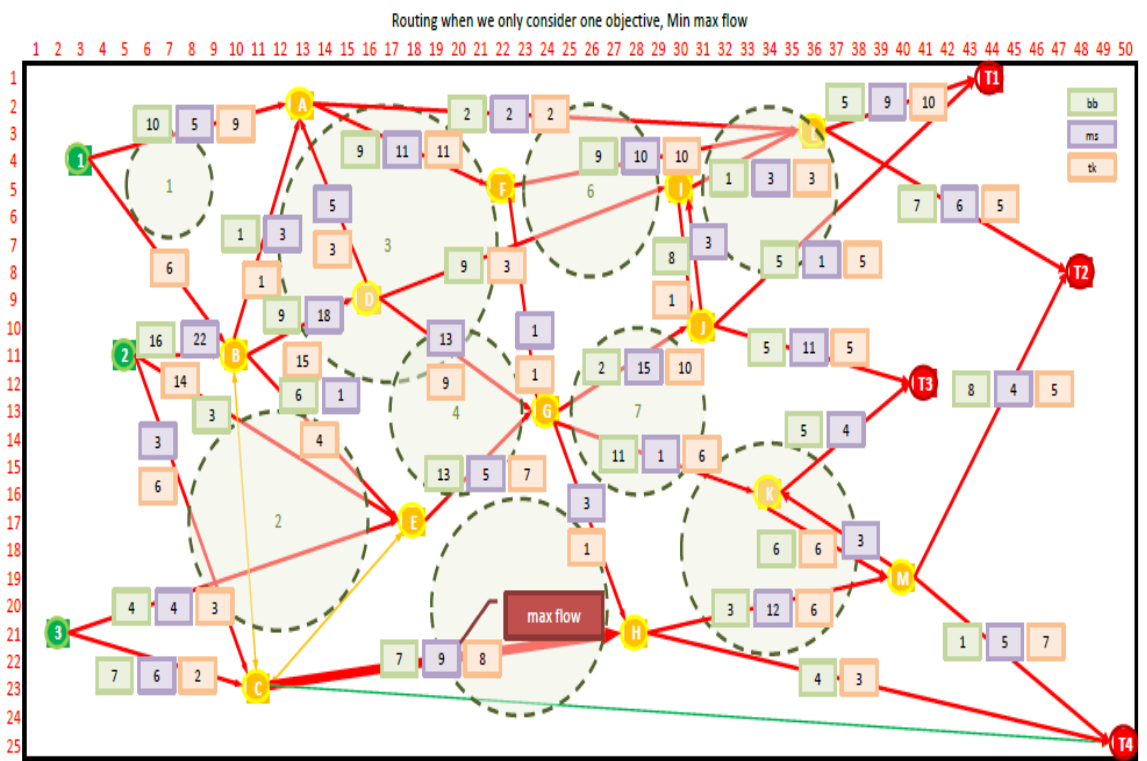


Figure 7. Route for min-max flow

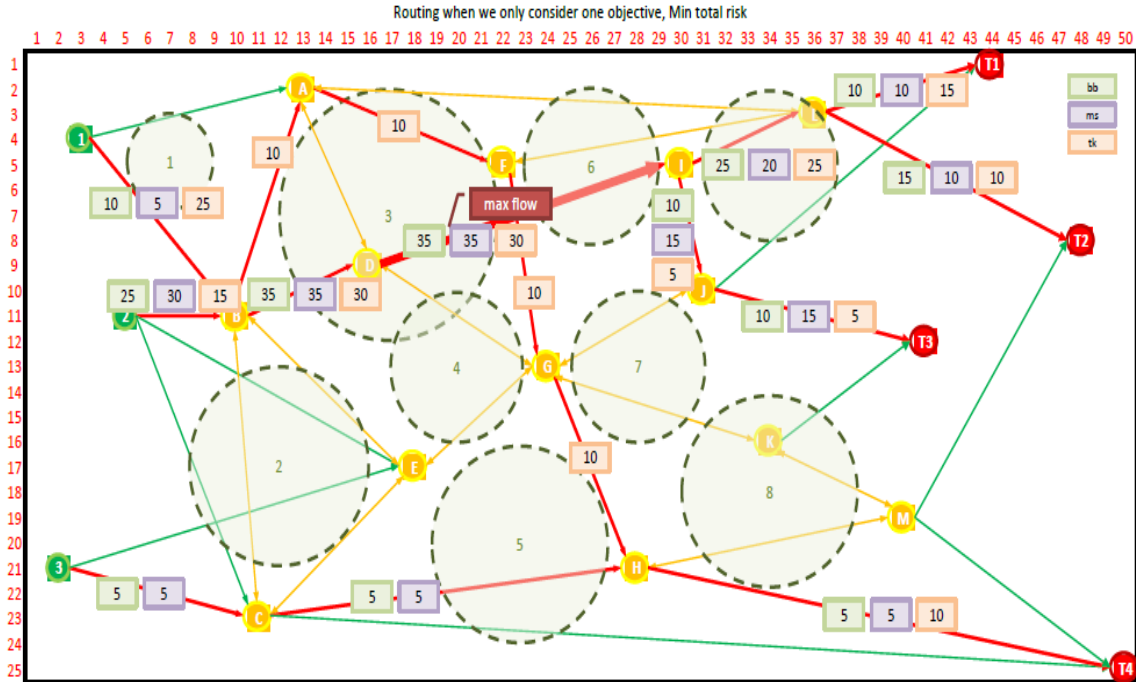


Figure 8. Route for minimum risk problem

- Weighted multiple objectives: equal weights

When we consider all three objectives at the same time, and their importance level is also same, the assigned elements on the arcs are shown in Figure 9. Using these numbers, we can get the objective value for min distance problem, min-max problem, and min risk problem. Their values are total distance of 5648, maximum flow of 467.2, and total risk of 1080. Those values are a little bigger than the objective value of each single objective problem.

- Weighted multiple objectives: more weight on total distance

When we consider all three objectives and minimizing total traveling distance is the most important objective, the assigned elements are shown in Figure 10. The solution result for total distance (5247.9) is bigger when we only consider minimizing total

distance problem (5135.5). Maximum flow (622.9) is smaller and total risk (1127.83) is slightly bigger than the case that we only consider the minimizing total traveling distance (maximum flow of 690 and total risk of 1120.85).

- Weighted multiple objectives: more weight on maximum flow

When we consider the most important objective among 3 objectives is minimizing maximum flow, the assigned elements are shown in Figure 11. In this case, solution result for maximum flow is 450.4, which is slightly bigger than objective value of minimizing maximum flow problem (437.1). Other solutions (total distance of 5593.3, total risk of 1159.21) are smaller than the case that we only consider the single objective problem (total distance of 5676.2, total risk of 1386.67).

- Weighted multiple objectives: more weight on risk

When minimizing total risk is the most important among 3 objectives, the assigned elements are shown in Figure 12. Using this number, we can get the solution for total risk as 922.98 and this value is slightly bigger than the case that we only consider the minimizing total risk (915.9). Maximum flow (876.4) is smaller and total distance (6302.8) is bigger than the case when we only consider minimizing total risk problem.

weighted values	w1=1, w2=1, w3=1	w1=0.8, w2=0.1, w3=0.1	w1=0.1, w2=0.8, w3=0.1	w1=0.1, w2=0.1, w3=0.8
total distance	5648	5247.9	5593.3	6302.8
maximum flow	467.2	622.9	450.4	876.4
total risk	1080	1127.83	1159.21	922.98

Table 4. Optimization result of weighted sum

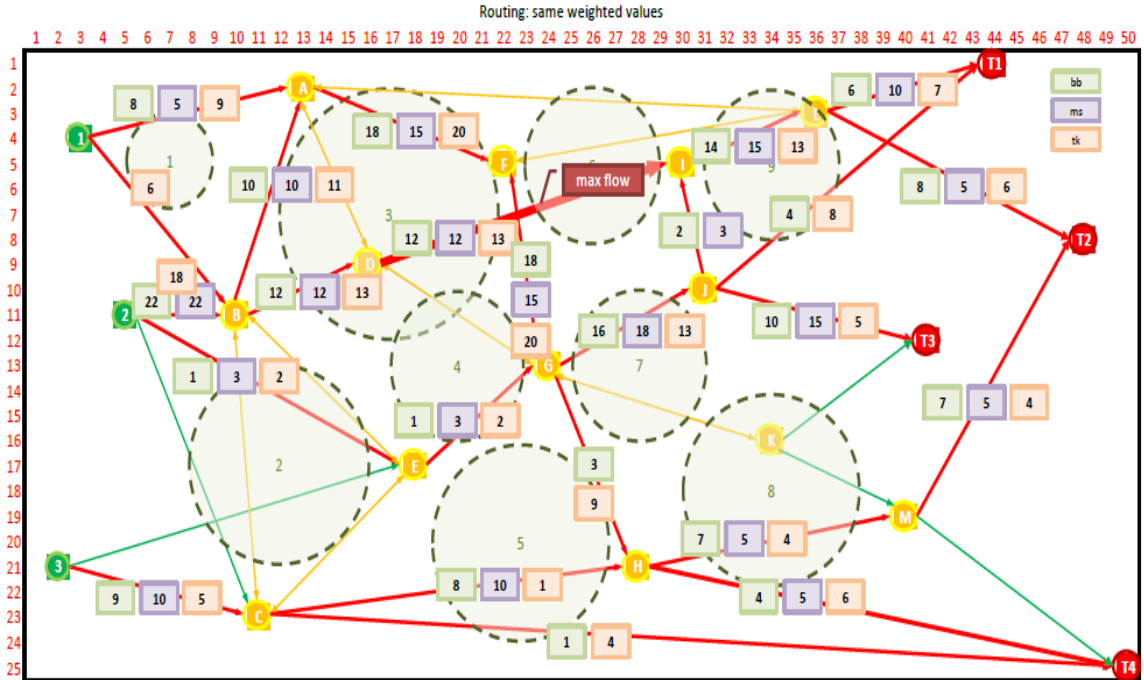


Figure 9.  $w_1 = 1, w_2 = 1, w_3 = 1$

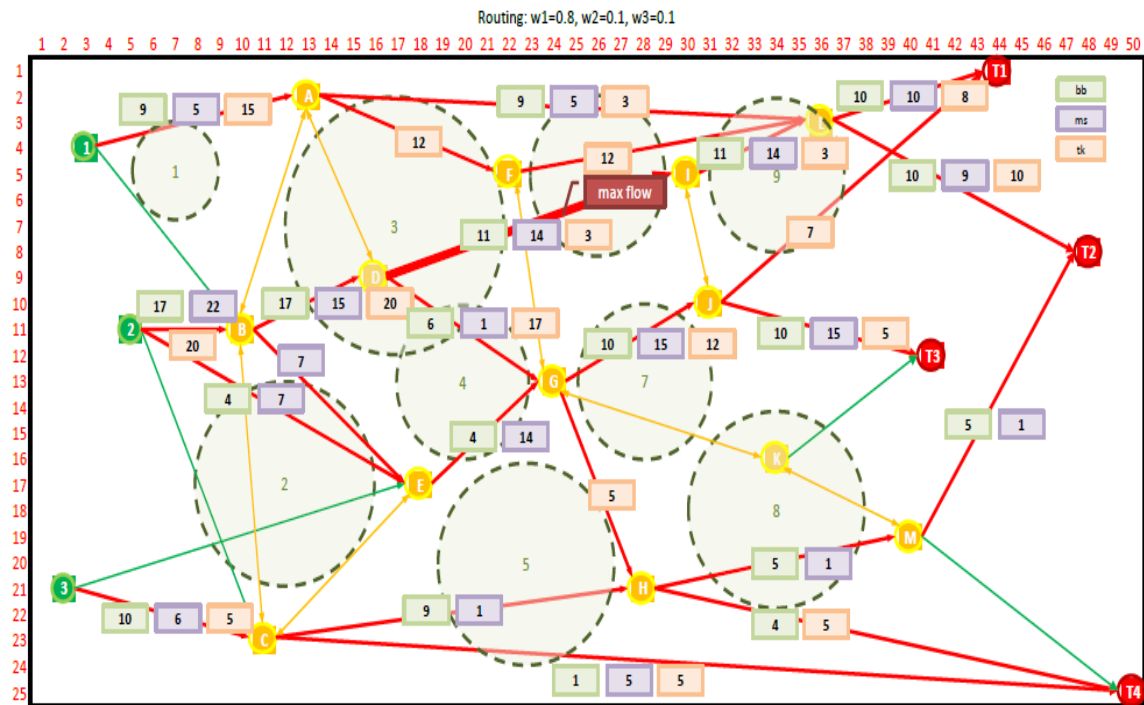


Figure 10.  $w_1 = 0.8, w_2 = 0.1, w_3 = 0.1$

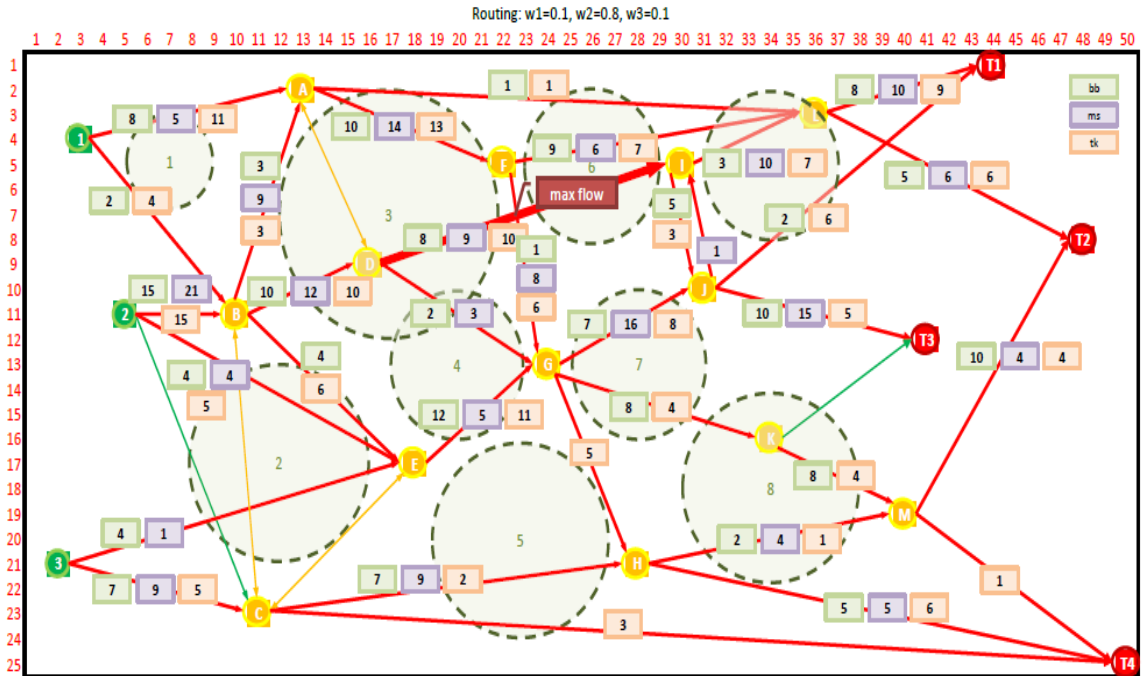


Figure 11.  $w_1 = 0.1, w_2 = 0.8, w_3 = 0.1$

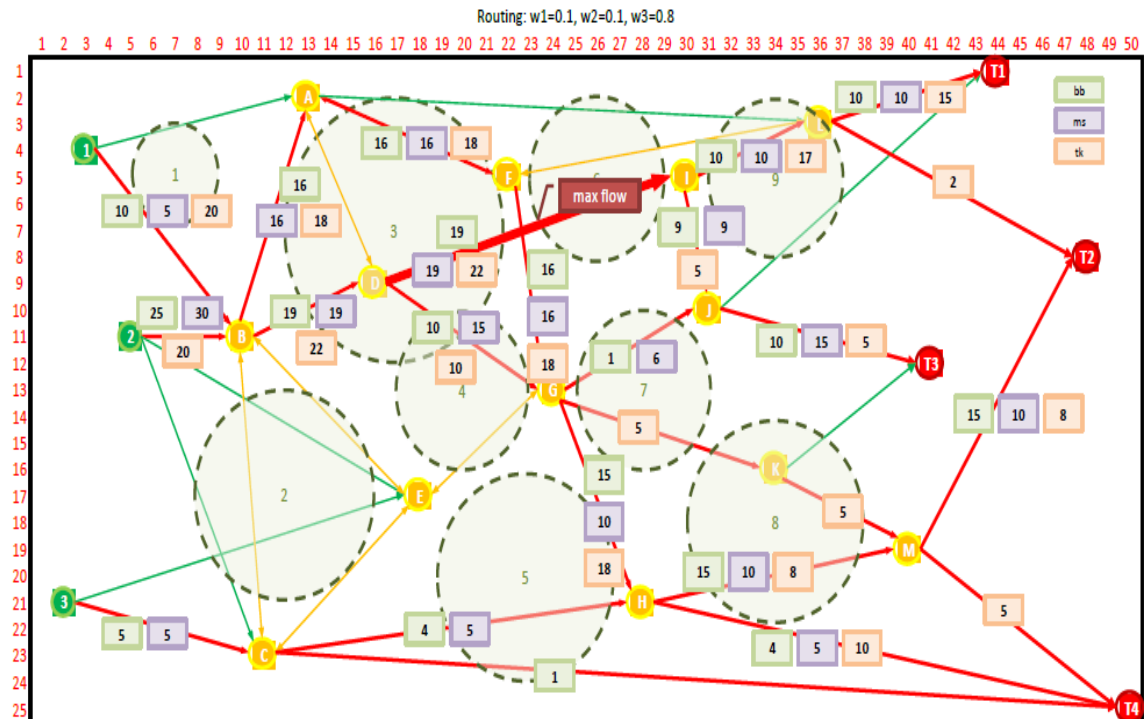


Figure 12.  $w_1 = 0.1, w_2 = 0.1, w_3 = 0.8$

## Goal programming

We first solve the single objective goal programming. Then, multiple objectives goal programming is combined with weighted values.

- Set goals (objective value of single objective problem \*1.1)
  - Total traveling distance goal: 5649
  - Min-Max flow problem: 481
  - Total risk goal:1007
- Solution for single goal programming: total distance goal =5649

When we set the minimizing total distance goal as 5649, the assigned elements are shown in Figure 13. This solution results in a total distance of 5648, maximum flow of 1075.8, and total risk of 1301.22. Compared to the solution sets of traditional single objective problem which is minimizing total distance, total distance, maximum flow, and total risk are bigger (5135.5, 690, 1120.85).

- Solution for single goal programming: maximum flow goal=481

When we set the goal for maximum flow as 481, the assigned elements are shown in Figure 14. This solution results in a maximum flow of 481, total distance of 5654.6, and total risk of 1347.09. Compared to the solution sets of traditional single objective problem which is minimizing maximum flow, maximum flow is slightly bigger (437.1) and total distance and total risk is smaller (5676.2, 1386.67).

- Solution for single goal programming: total risk goal=1007

When we set the goal for minimizing total risk problem as 1007, the assigned elements are shown in Figure 15. This solution results in a total risk of 1007, total

distance of 5564.2, and maximum flow of 1109.6. Compared to the solution sets of traditional single objective problem which is minimizing total risk, total distance and total risk are slightly bigger (5550, 915.9), and maximum flow is smaller (1460).

objective	total distance goal: 5649	maximum flow goal: 481	total risk goal: 1007
total distance	5648	5654.6	5564.2
maximum flow	1075.8	481	1109.6
total risk	1301.22	1347.09	1007

Table 5. Result of single objective goal programming

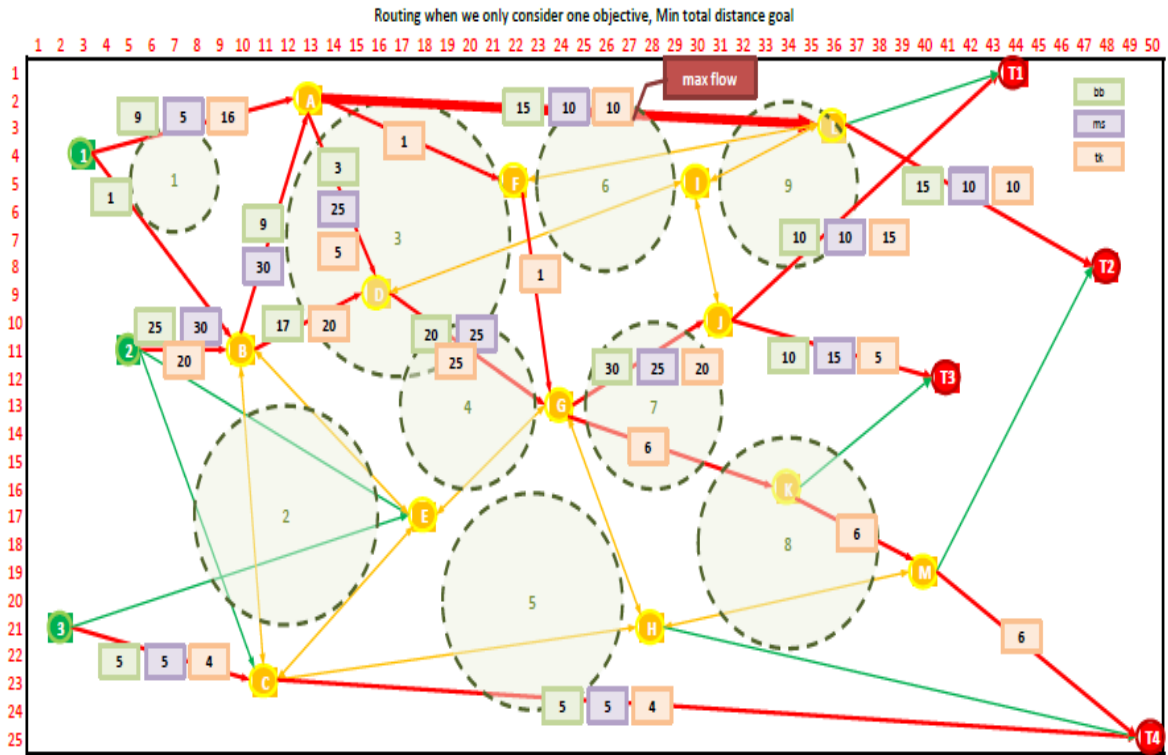


Figure 13. Total traveling distance goal: 5649





- Solution for multi-objective goal programming: equal weights

When we set the goals for each objective as 5649 (total distance goal), 481 (maximum flow goal), and 1007 (total risk goal) and weighted values for all objectives are same, the assigned elements are shown in Figure 16. When we apply these numbers to the original objectives, we can get 5648 for total distance (compared to weighted sum result: 5648), 499 for maximum flow (weighted sum result: 467.2), and 1080 for total risk (weighted sum result: 1080).

- Solution for multi-objective goal programming: more weights on distance

When our goal is same as above problem and weighted value of each goal is 0.8 for total distance goal, 0.1 for maximum flow goal and 0.1 for total risk goal, the assigned elements are shown in Figure 17. When we apply these numbers to the original objectives, we can get 5641 for total distance (compared to equally weighted goal programming result: 5247.9), 495 for maximum flow (compared to equally weighted goal programming result: 622.9), and 1101 for total risk (compared to equally weighted goal programming result: 1159.21).

- Solution for multi-objective goal programming: more weights on maximum flow

When our goal is same as above problem and weighted value of each goal is 0.1 for total distance goal, 0.8 for maximum flow goal and 0.1 for total risk goal, the assigned troops are shown in Figure 18. When we apply these numbers to the original objectives, we can get 5649 for total distance (compared to equally weighted goal programming result: 5593.3), 481 for maximum flow (compared to equally weighted goal programming

result: 450.4), and 1101 for total risk (compared to equally weighted goal programming result: 1159.21).

- Solution for multi-objective goal programming: more weights on risk

When our goal is same as above problem and weighted value each goal is 0.1 for total distance goal, 0.1 for maximum flow goal and 0.8 for total risk goal, the assigned troops are shown in Figure 19. When we apply these numbers to the original objectives, we can get 5651 for total distance (compared to equally weighted goal programming result: 6302.8), 680 for maximum flow (compared to equally weighted goal programming result: 876.4), and 1007 for total risk (compared to equally weighted goal programming result: 922.98).

weighted values	$w_1=1, w_2=1, w_3=1$	$w_1=0.8, w_2=0.1, w_3=0.1$	$w_1=0.1, w_2=0.8, w_3=0.1$	$w_1=0.1, w_2=0.1, w_3=0.8$
total distance	5648	5641	5649	5651
maximum flow	499	495	481	680
total risk	1080	1084	1101	1007

Table 6. Result of multi objective goal programming

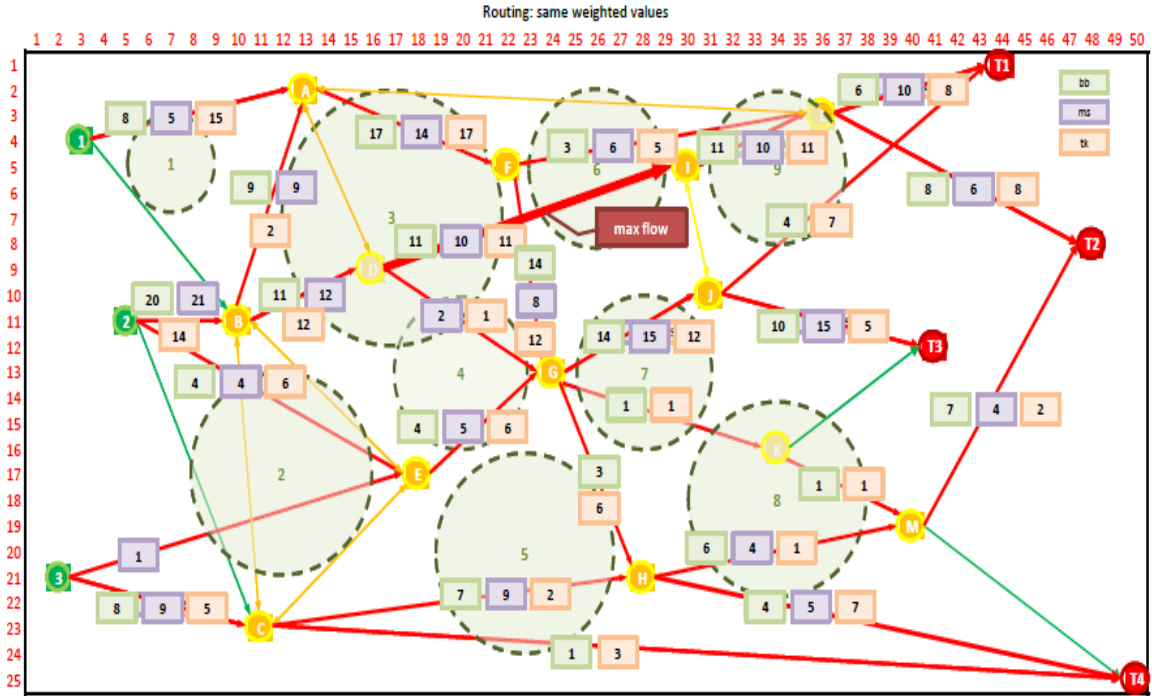


Figure 16.  $w_1 = 1, w_2 = 1, w_3 = 1$  for each goal

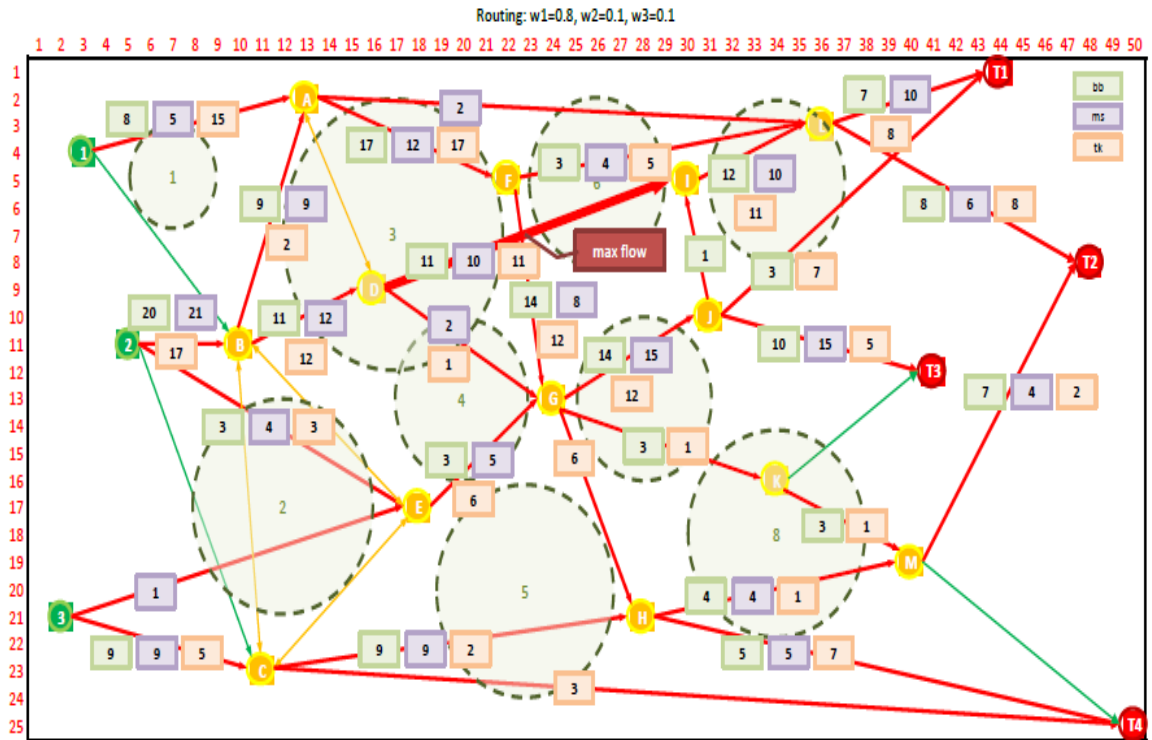


Figure 17.  $w_1 = 0.8, w_2 = 0.1, w_3 = 0.1$  for each goal

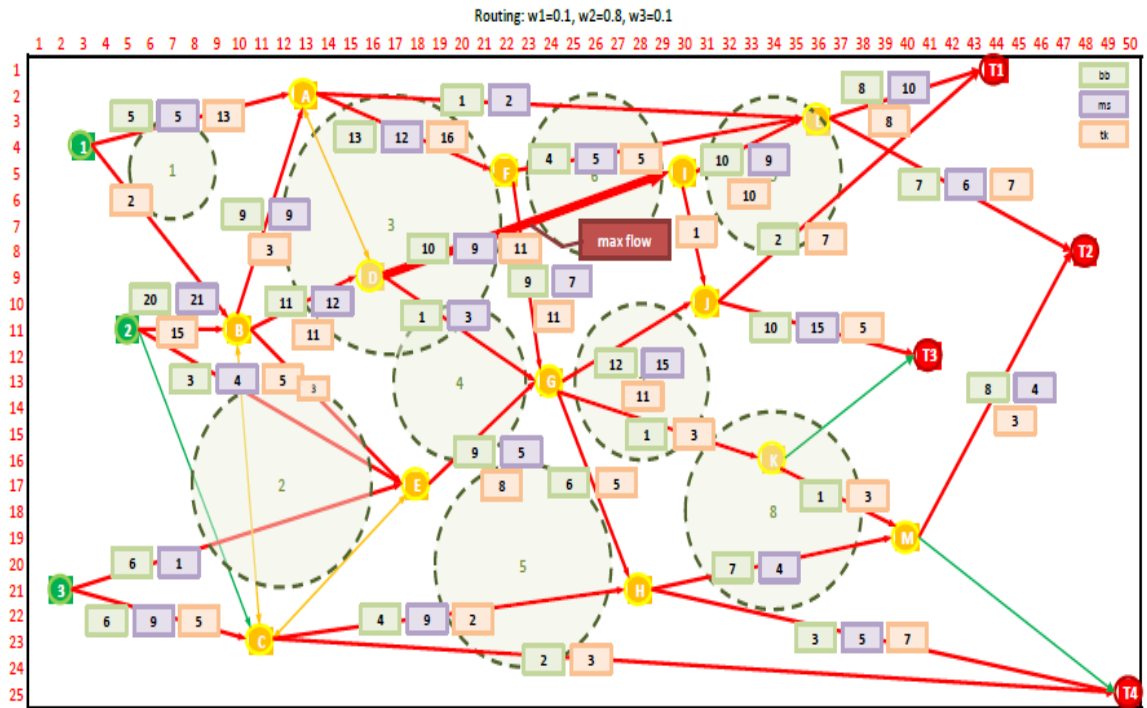


Figure 18.  $w_1 = 0.1, w_2 = 0.8, w_3 = 0.1$  for each goal

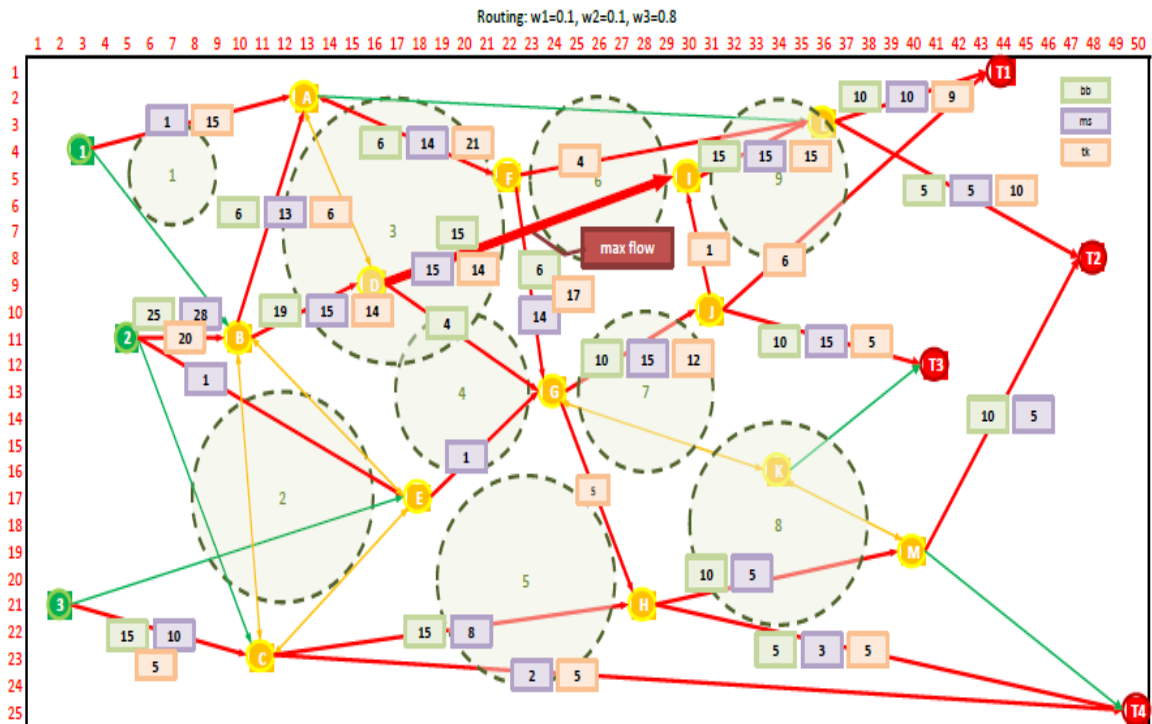


Figure 19.  $w_1 = 0.1, w_2 = 0.1, w_3 = 0.8$  for each goal

## Lexicographic goal programming

- Goal setting (parameter setting)

- $f_j(x) \leq T_j$  where  $T_j = (1 + \alpha_j) * f_j(x_j^*)$

- 1<sup>st</sup> priority goal:  $\alpha_1 = 0.05$

This means that when we solve the 2<sup>nd</sup> priority objective problem, our first priority objective goal is at most 5% greater than the objective value of first priority problem.

- 2<sup>nd</sup> priority goal:  $\alpha_2 = 0.1$

This means that when we solve the 3<sup>rd</sup> priority objective problem, our second priority objective goal is at most 10% greater than the objective value of second priority problem.

- Priorities: min total risk > min total distance > min maximum flow

- Result of minimizing total risk

When we consider the risk minimization is the most important factor among 3 objectives, we first optimize the minimum risk problem. Since the objective value is 915.9, we set the goal for min risk problem as 962.

- Result of minimizing total distance constrained with 1<sup>st</sup> priority goal

After setting 1<sup>st</sup> priority goal, we optimize the 2<sup>nd</sup> priority problem, minimizing total distance. The assigned numbers of elements for some paths are changed and solution of total risk is also changed (915.9 to 961.87) after the optimizing process. Since we set the total risk goal as 962, the solution of total distance (5263.6) is slightly

bigger than single objective problem which is minimizing total distance (5135.5). For the final optimization, we set the total distance goal as 5790.

- Result of minimizing maximum flow constrained with 1<sup>st</sup> and 2<sup>nd</sup> priority goals

After setting 1<sup>st</sup> and 2<sup>nd</sup> goals, we optimize the minimizing maximum flow. Because of new objective and constraints, the assigned numbers of elements for some paths are changed and the solution of total risk (961.87 to 961.83) and total distance (5263.6 to 5786.4) are slightly changed.

- Route (after min-max flow optimization)

After 3<sup>rd</sup> optimization, the suggested route is below picture and total traveling distance is 5786.4, total traveling risk is 961.93 and maximum flow is 803.

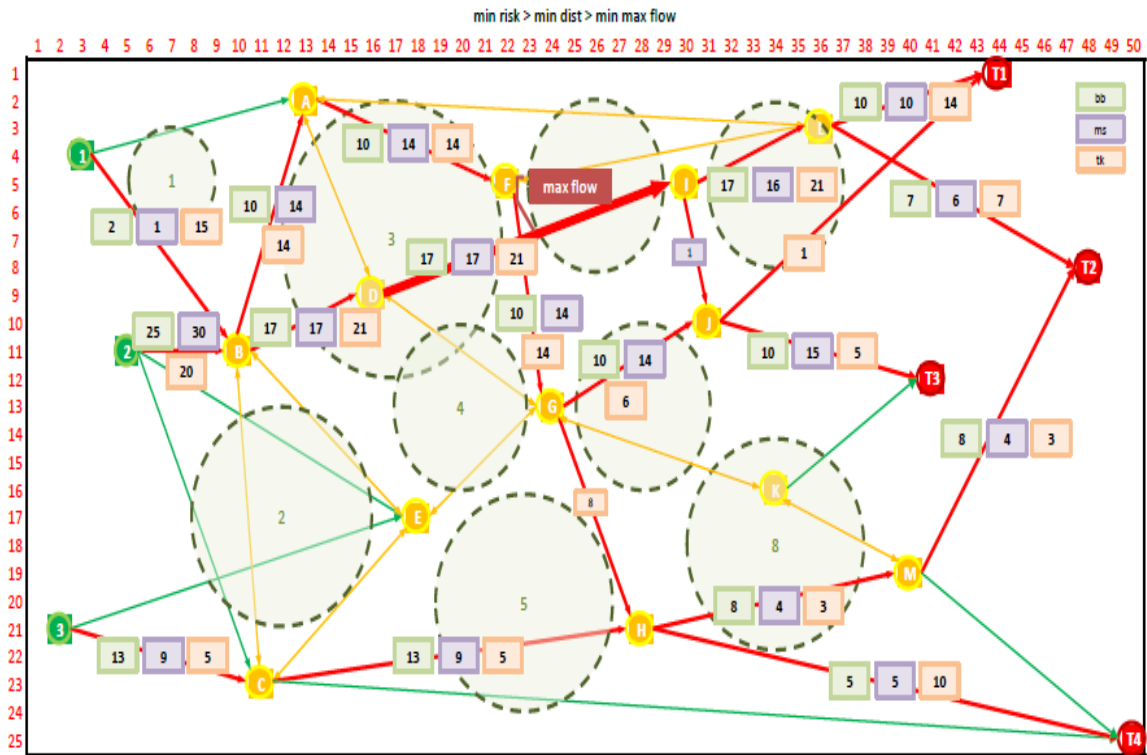


Figure 20. priority: min risk > min distance > min-max flow

- Result table and value changes

min total risk > min total distance > min max flow			
order	total distance	maximum flow	total risk
1st priority	5550	1460	<b>915.9</b>
2nd priority	<b>5263.6</b>	1022	961.87
3rd priority	5786.4	<b>803</b>	961.93

Table 7. Results of optimization process (min risk > min distance > min-max flow)

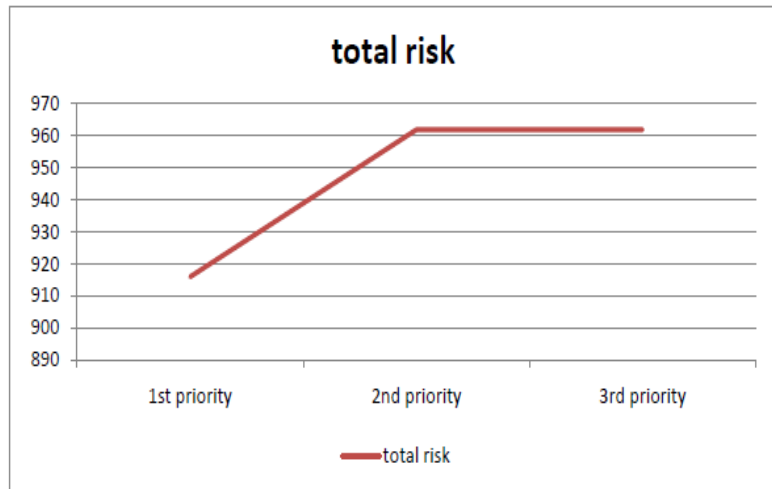


Figure 21. Total risk changes (priority: min risk > min distance > min-max flow)

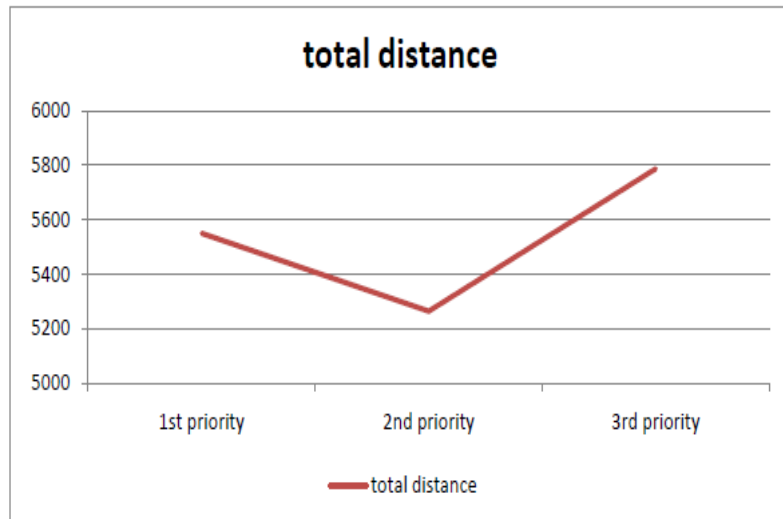


Figure 22. Total distance changes (priority: min risk > min distance > min max flow)

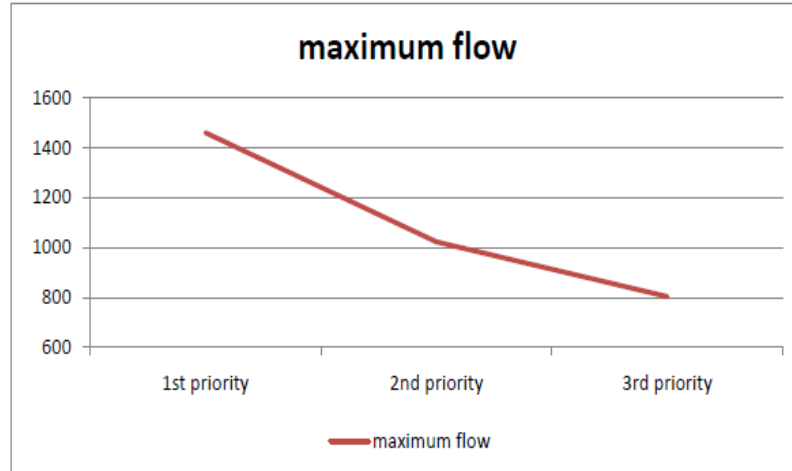


Figure 23. Max flow changes (priority: min risk > min distance > min-max flow)

- Priorities: min total distance > min maximum flow > min risk
  - Result of minimizing total distance

When we consider the minimizing total traveling distance is most significant problem among 3 objectives, the first optimization result is as follow. Since the objective value is 5135.5, we set the distance goal for 2<sup>nd</sup> priority problem as 5392.

- Result of minimizing maximum flow constrained with 1<sup>st</sup> priority goal

After setting 1<sup>st</sup> priority goal, we optimize the minimizing maximum flow. The assigned elements for some paths are changed and total distance value is also changed (5135.5 to 5391.6) after the optimizing process. In this case, the result of maximum flow is decreased from 690 to 453.6, so we set the maximum flow goal as 499.

- Result of minimizing total risk constrained with 1<sup>st</sup> and 2<sup>nd</sup> priority goals

After setting total distance goal and maximum flow goal, we optimize minimizing total risk. Because of new objective and constraints, the numbers of assigned elements



for some paths are changed and total distance (5391.6 to 5391.9) and maxim flow (453.6 to 499) are also changed.

- Route (after min risk optimization)

After 3<sup>rd</sup> optimization, the suggested route is below picture and total traveling distance is 5391.9, total traveling risk is 1134.59 and maximum flow is 499.

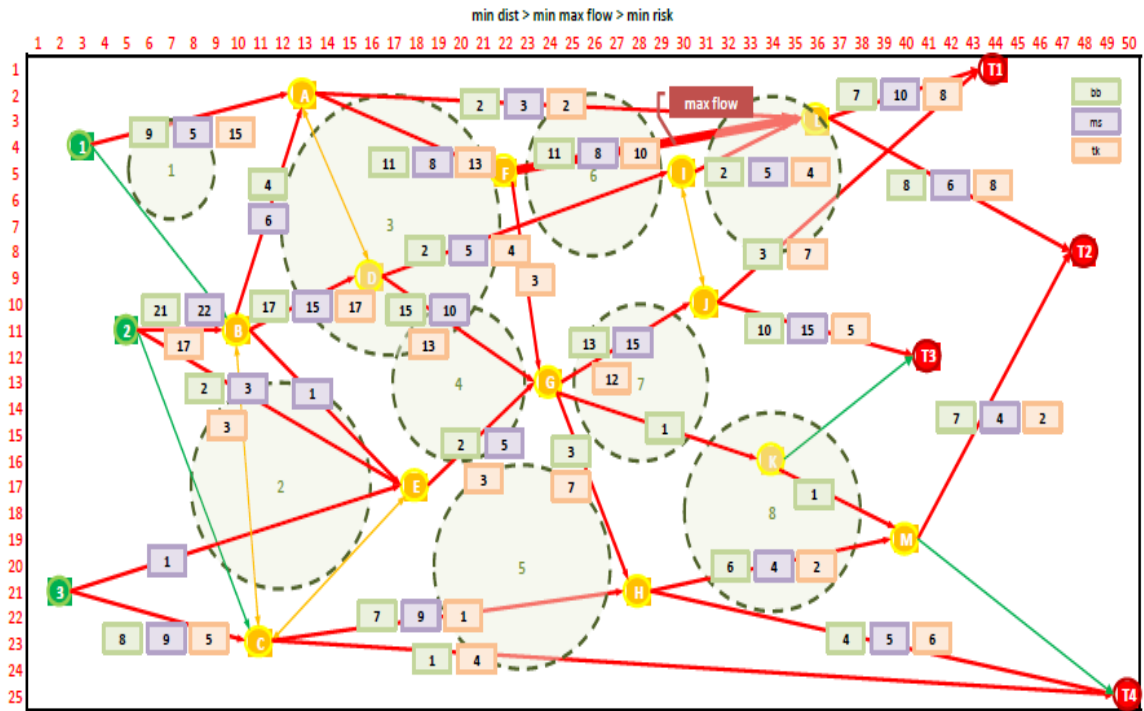


Figure 24. priority: min distance > min-max flow > min risk

- Result table and value changes

min total distance > min max flow > min total risk			
order	total distance	maximum flow	total risk
1st priority	5135.5	690	1120.85
2nd priority	5391.6	453.6	1226.03
3rd priority	5391.9	499	1134.59

Table 8. Results of optimization process (min distance > min-max flow > min risk)

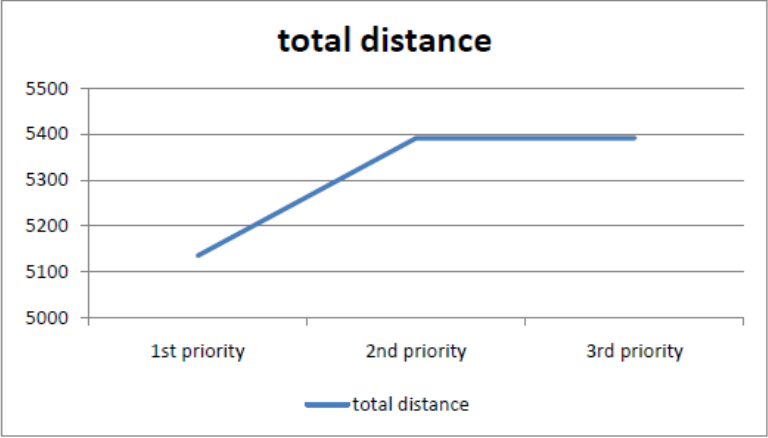


Figure 25. total distance value change (priority: min distance > min-max flow > min risk)

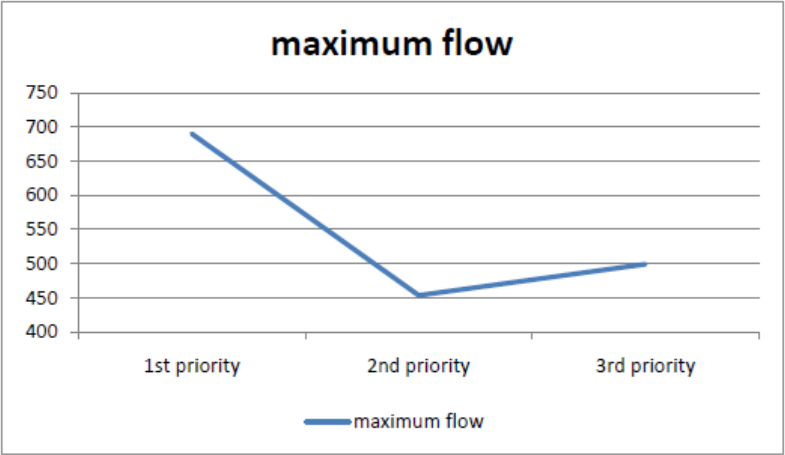


Figure 26. max flow value change (priority: min distance > min-max flow > min risk)

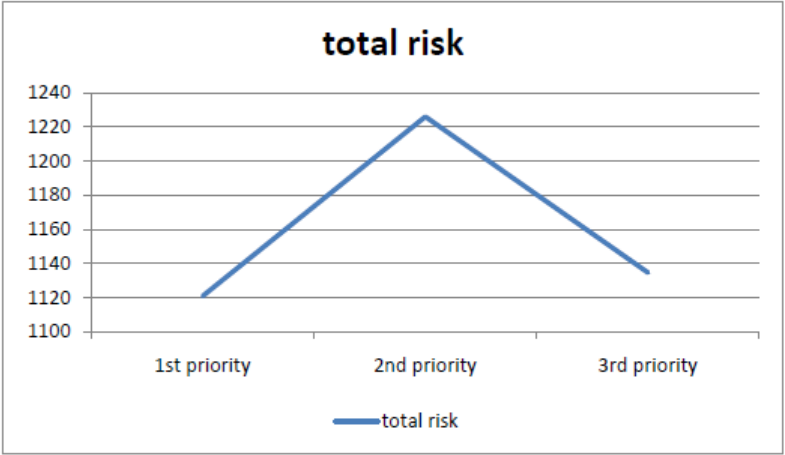


Figure 27. total risk value change (priority: min distance > min-max flow > min risk)

- Priorities: min-max flow > risk minimization > distance minimization
  - Result of minimizing maximum flow

When we consider the min-max flow problem is the most important among 3 objectives, the result is as follow. Since the object value is 437.1, we set the min-max flow goal for 2<sup>nd</sup> priority optimization as 459.

- Result of min risk problem constrained to 1<sup>st</sup> priority goal

After setting 1<sup>st</sup> priority goal, we optimize minimizing total risk. The assigned numbers of elements for some paths are changed, so maximum flow is also changed (437.1 to 459). The result of total risk is 1126.47, so risk goal for final optimization process, min distance problem, as 1239.

- Result of min distance problem constrained to 1<sup>st</sup> and 2<sup>nd</sup> priority goals

After setting maximum flow goal and total risk goal, we optimize minimizing total distance. Because of new objective and constraints, the numbers of assigned elements for some paths are changed and total risk (1126.47 to 1225.72) is also changed.

- Result table and value changes

min max flow > min total risk > min total distance			
order	total distance	maximum flow	total risk
1st priority	5676.2	<b>437.1</b>	1386.67
2nd priority	5743.3	459	<b>1126.47</b>
3rd priority	<b>5374.5</b>	459	1225.72

Table 15. Results of optimization process (min-max flow > min risk > min distance)

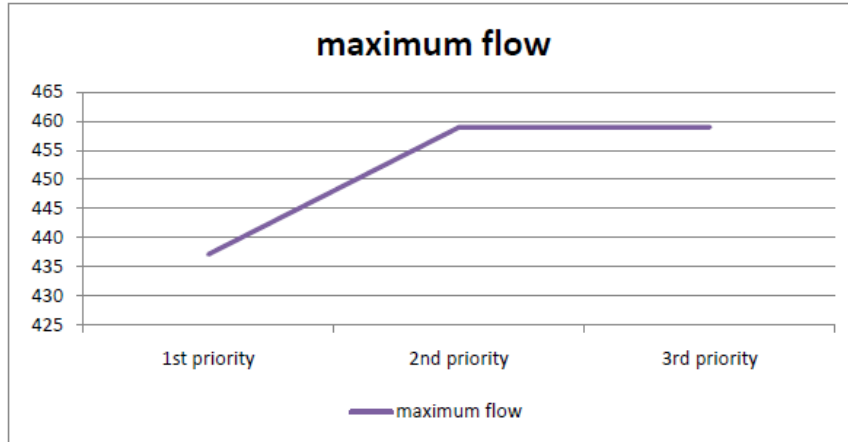


Figure 28. max flow value change (priority: min-max flow > min risk > min distance)

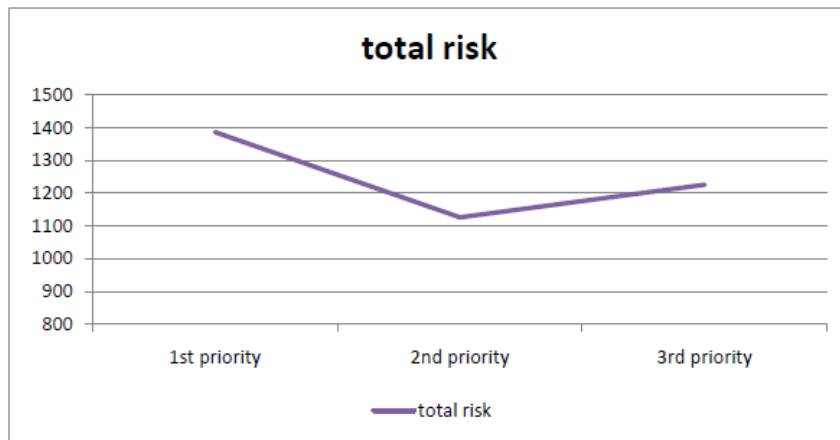


Figure 29. total risk value change (priority: min-max flow > min risk > min distance)

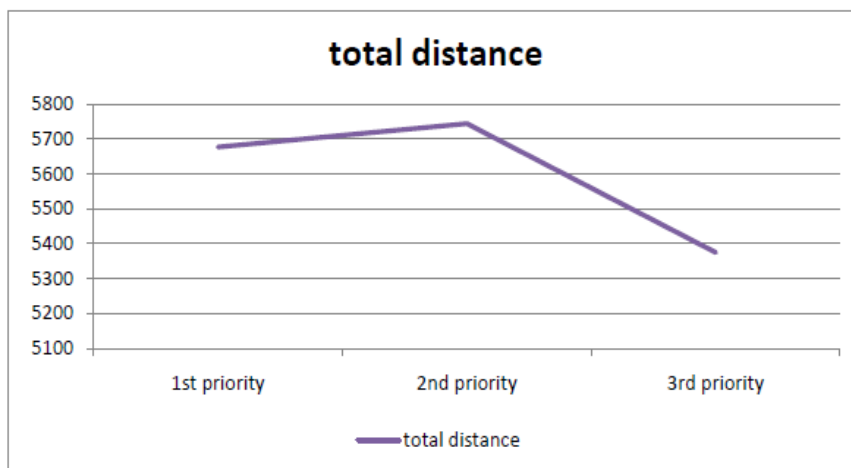


Figure 30. total distance value change (priority: min-max flow > min risk > min distance)

- Route (after min distance optimization)

After 3<sup>rd</sup> optimization, the suggested route is below picture and total traveling distance is 5374.5, total traveling risk is 1225.71 and min-max flow is 459.

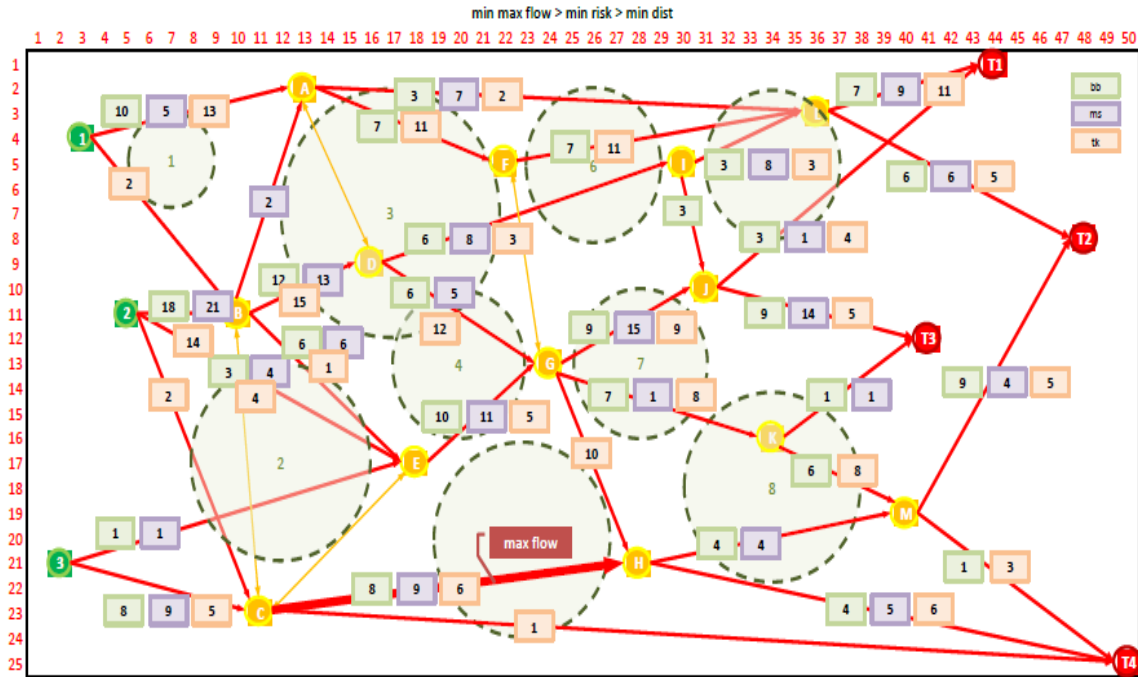


Figure 31. priority: min-max flow > min risk > min distance

- Experimental result analysis (see Appendix for full result table)

As the above pictures, the results show various solution sets when we consider 3 objectives in one model based on the preferences. There also exist some differences among the solution approaches like number of elements assigned in a certain path and paths that the elements will use. However, the general patterns are similar such as when the weighted values are increasing, objective values are decreasing because all objective functions are minimizing problem and there exist the avoidance of risk paths, especially when there existed alternative routes, and usage of diversified (or various) paths.

## 2.6 Conclusion

In this chapter, we have discussed about the multi-objective routing problem with integer programming. We have considered three objectives: minimizing total traveling distance, minimizing maximum flow, and minimizing total traveling risk based on the traditional transportation network problem. To solve the problem, we have 3 different types of solution approach: weighted sum approach, goal programming, and lexicographical goal programming. There are lots of solution sets based on the importance levels or priorities for our objectives, so we can pick the route that is suitable for a certain condition.

# CHAPTER 3. DYNAMIC ROUTING IN A DYNAMICALLY CHANGING HOSTILE ENVIRONMENT: MODELS AND ALGORITHMS

---

## 3.1 Problem description

In chapter 2, the locations of enemies (i.e. potential IED attack points) are assumed to be static to simplify the calculation of risk. However, in reality, location of hostile threats changes dynamically and “intelligently” in order to maximize damage to mobile troops. In this chapter, we consider the dynamic nature of intelligence about the enemy activities and develop routing policies many intelligence feedback. In the battlefield, one can obtain about enemy activities from satellite or other sources, and such

information can be extremely useful to guide a troop's movements in a hostile environment.

We model the dynamic routing problem in a battlefield as a Markov decision process. We assume that there is a single troop that needs to go from a given location to a specified destination and we consider two main objectives: minimizing total travel distance and minimizing risk of encountering with a hostile entity.

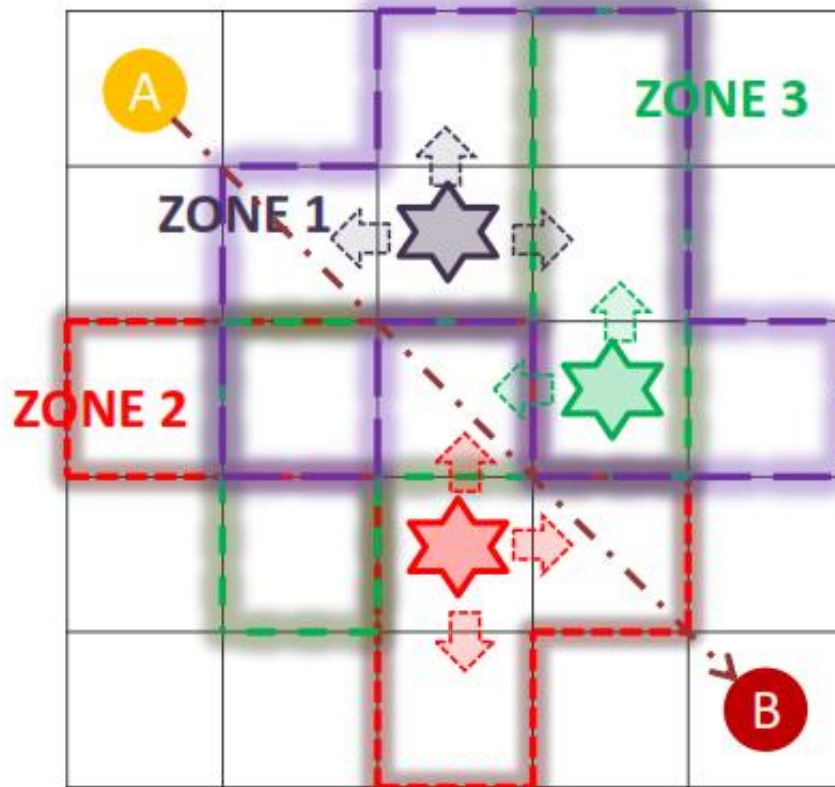


Figure 32. Simple map for MDPs



Figure 23 illustrates the dynamic battlefield routing problem. Our objective is to move from A to B with minimum cost which includes the movement cost and encounter cost. Every time the troop takes a step, the movement cost is incremented by one. Every time the troop encounters with an enemy, enemy cost is incremented by a constant value that represents risk of engaging with that particular enemy.

We assume that each party (the troop and the enemies) can move at most one step at a time, and there are nine possible movement actions they can take: staying at the same location or moving north (N), south (S), east (E), west (W), northeast (NE), northwest (NW), southeast (SE), southwest (SW). Each enemy has is assumed to have an action zone outside of which it cannot move. Lastly, the battlefield terrain is modeled as a two-dimensional grid and the altitude or ground configuration are neglected on this thesis.

This chapter is organized as follows. In section 3.2, we review the literatures. In section 3.3, 3.4 and 3.5, we discuss about our Markov decision model and its solution approaches. In section 3.6, we show the result of small size problem.

### 3.2 Literature review

Orienteering is a kind of outdoor sports, which requires navigation skills using map and compass and its usual aim is to find a shortest path from one point to another. Hayes and Norman (1984) use a dynamic programming framework for this sports and their model is based on a grid map. Possible movement directions from one point are eight adjacent points and there are different costs associated with different movements due to the terrain.

In operations research, two solution methods are given for the shortest path routing problem by Pollak and Wiebenson (1959): one is the shortest route tree and the other is matrix method based on the dual solution of the shortest path routing problem. In the matrix method, they used a dynamic programming approach. Derman (1962) introduced linear programming to derive optimal rules for control systems. The author defined that system is controlled by making one of finite decisions and the decisions affect the future operations in the system.

Psaraftis (1995) focused on the importance of the dynamic vehicle routing problem because of the real time availability of information. In his report, the author discussed various aspects of the vehicle routing problem and solution methodologies. Dror, Laporte, and Trudeau (1989) modeled the vehicle routing problem with stochastic demands as a Markov decision process. Aberdeen, Thièbaux, and Zhang (2004) also used a Markov decision model for the military operations planning problem and solved it using the value iteration algorithm.

### 3.3 A Markov Decision Model

#### Notations and parameters

- $\mathcal{N}$ : Set of points in a grid or a discretized network, where the distance between two adjacent points is one unit distance.  $\mathcal{A}(s)$ : Set of points adjacent to point  $s \in \mathcal{N}$ .
- $h$ : Number of hostile entities. Let  $\mathcal{H}_i \subset \mathcal{N}$  be the set of points that represent the activity area of hostile entity  $i, i = 1, 2, \dots, h$ .
- $d$ : Cost of moving one unit distance.
- $e_i$ : Cost of encountering with a hostile entity  $i, i = 1, 2, \dots, h$ .
- $\gamma$ : Discount rate.

#### MDP

- $S$ : state which is a vector representing the location of all parties

As described above, all the information related to the locations are coordinate points of each party.

$s_0 \in \mathcal{N}$  is the troops' location and  $s_i \in \mathcal{H}_i, i = 1, 2, \dots, h$ , are the location of the hostile entities. In vector form,

$$s = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \\ \vdots \\ s_h \end{bmatrix}.$$

Let  $\mathcal{S}$  be the set of states. Note that  $|\mathcal{S}| = |\mathcal{N}| \times |\mathcal{H}_1| \times |\mathcal{H}_2| \times \dots \times |\mathcal{H}_h|$ .

- $x \in \mathcal{A}(s_0) \cup \{s_0\}$  is the action representing where we move next.
- Random disturbance and system dynamics:

The hostile entities randomly move from one location to another at every decision epoch. We denote by  $\mathcal{W}_{t+1}$  the random movements of enemies that become known at time  $t + 1$ . Letting  $s_t$  and  $x_t$  are the state vector and action, respectively, at time  $t$ , we use the following notation to represent the state transition:

$$s_{t+1} = \Gamma(s_t, x_t, \mathcal{W}_{t+1}).$$

We assume that the probability distribution of the random variable  $\mathcal{W}_{t+1}$  is known. That is, the one-step transition probabilities  $Prob\{s_{t+1}|s_t, x_t\}$  are known.

- DP recursion:

The Bellman equation can be written as

$$\mathcal{V}(s) = \min_{x \in \mathcal{A}(s_0) \cup \{s_0\}} \left\{ A(s, x) + \gamma \sum_{s' \in \mathcal{S}} prob\{s'|s, x\} * \left( E(s') + \mathcal{V}(s') \right) \right\},$$

where  $A(s, x)$  is the action cost and  $E(s')$  is the encounter cost.

### 3.4 Transition Probability and Optimal Action

#### Basic rule for transition probabilities

The transition probability value for a certain action during time  $t$  is related to the information of enemies' next actions.

- The troop and the enemies know the location of each party at the beginning of time  $t$ .

We make this assumption is because the given state at the beginning of time  $t$  ( $S_t$ ) contains all information about current locations.

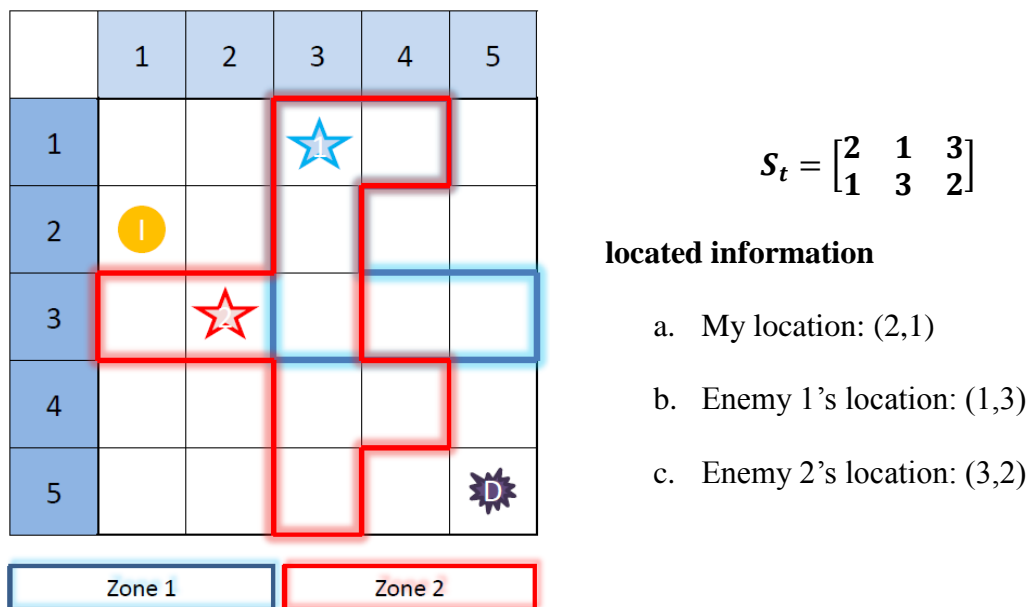
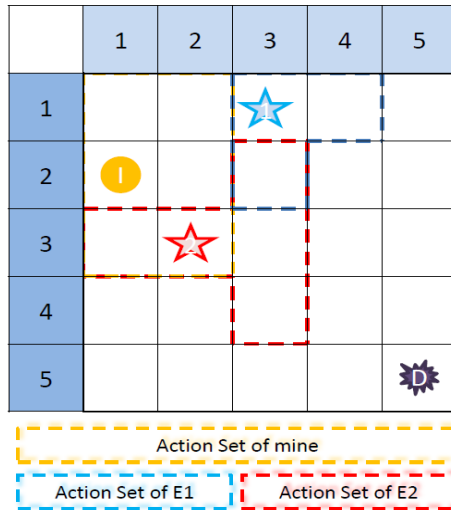


Figure 33. State information at the beginning of time  $t$

- Each party can predict the opponents' possible action during time t. That is, a troop has information of the enemies' territories at the beginning stage.



$$s_t = \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}$$

**Possible Action Set during time t**

- a. A troop actions:  
(1,1) (1,2) (2,1) (2,2) (3,1) (3,2)
- b. Enemy 1's actions:  
(1,3) (1,4) (2,3)
- c. Enemy 2's actions:  
(2,3) (3,1) (3,2) (3,3) (4,3)

Figure 34. Possible action sets during time t

- Aggressiveness of Enemy

We know that enemies have a tendency to move toward the shortest distance point from the current location of the troop during time t. Moreover, each enemy has its own aggressive property toward to the troop.

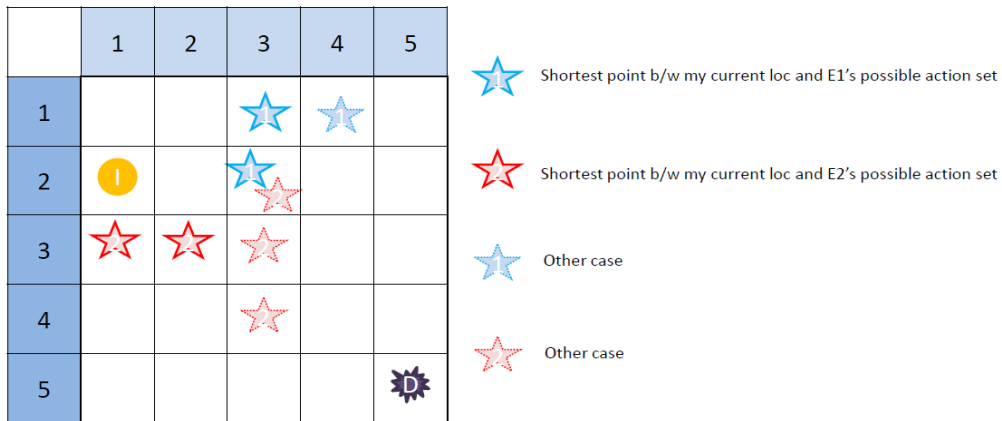


Figure 35. enemy's individual movement

- Aggressiveness of each enemy
  - Aggressiveness of Enemy 1: 1.8
  - Aggressiveness of Enemy 2: 1.5
- Probability value of each enemy during time t
  - Enemy 1:

$\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$  represents locations of the troop ( $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ) and enemy 1 ( $\begin{bmatrix} 1 \\ 3 \end{bmatrix}$ ) at the beginning of time t and  $prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\}$  means probability value of enemy 1's action is staying, (1,3), during time t when the current located information is  $\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}$ .

$$prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} = \frac{1.8}{1.8 + 1.8 + 1} \approx 0.39$$

$$prob\{(1,4)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} = \frac{1}{1.8 + 1.8 + 1} \approx 0.22$$

$$prob\{(2,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} = \frac{1.8}{1.8 + 1.8 + 1} \approx 0.39$$

- Enemy 2:

$\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}$  represents locations of the troop ( $\begin{bmatrix} 2 \\ 1 \end{bmatrix}$ ) and enemy 2 ( $\begin{bmatrix} 3 \\ 2 \end{bmatrix}$ ) at the beginning of time t and  $prob\{(3,1)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\}$  means probability value of action of enemy 2 is moving to (1,3) during time t when the current located information is  $\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}$ .

$$prob\{(3,1)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} = \frac{1.5}{1.5 + 1.5 + 1 + 1 + 1} = 0.25$$

$$prob\{(3,2)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} = \frac{1.5}{1.5 + 1.5 + 1 + 1 + 1} = 0.25$$

$$prob\{(3,3)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} = \frac{1}{1.5 + 1.5 + 1 + 1 + 1} \approx 0.17$$

$$prob\{(2,3)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} = \frac{1}{1.5 + 1.5 + 1 + 1 + 1} \approx 0.17$$

$$prob\{(4,3)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} = \frac{1}{1.5 + 1.5 + 1 + 1 + 1} \approx 0.17$$

- Independent movements of enemies

To simplifying the calculation, we assumed that each enemy's movement is independent from others. That is, in order to calculate the transition probability value during time t, we just need to multiply the probability value of each enemy movement during time t.

$prob\left\{\left(\begin{matrix} (1,3) \\ (3,1) \end{matrix}\right) \middle| \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}\right\}$  means that probability value of action sets of enemy1

and enemy 2 is (1,3) and (3,1) when current state is  $\begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}$ .

$$prob\left\{\left(\begin{matrix} (1,3) \\ (3,1) \end{matrix}\right) \middle| \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}\right\} = prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} * prob\{(3,1)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} \approx 0.098$$

$$prob\left\{\left(\begin{matrix} (1,3) \\ (3,2) \end{matrix}\right) \middle| \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}\right\} = prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} * prob\{(3,2)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} \approx 0.098$$

$$prob\left\{\left(\begin{matrix} (1,3) \\ (3,3) \end{matrix}\right) \middle| \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}\right\} = prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} * prob\{(3,3)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} \approx 0.065$$

$$prob\left\{\left(\begin{matrix} (1,3) \\ (2,3) \end{matrix}\right) \middle| \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}\right\} = prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} * prob\{(2,3)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} \approx 0.065$$

$$prob\left\{\left(\begin{matrix} (1,3) \\ (4,3) \end{matrix}\right) \middle| \begin{bmatrix} 2 & 1 & 3 \\ 1 & 3 & 2 \end{bmatrix}\right\} = prob\{(1,3)|\begin{bmatrix} 2 & 1 \\ 1 & 3 \end{bmatrix}\} * prob\{(4,3)|\begin{bmatrix} 2 & 3 \\ 1 & 2 \end{bmatrix}\} \approx 0.065$$



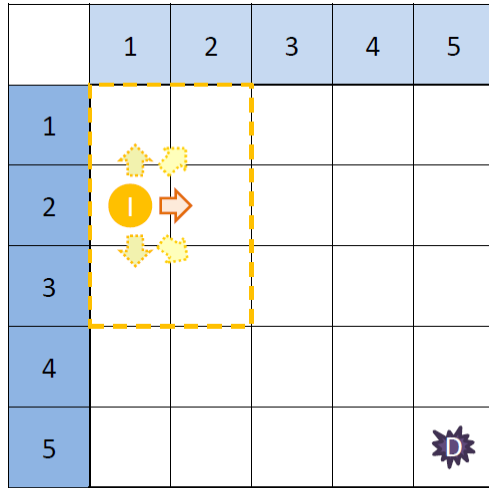



## A simple policy for the troop

At the beginning of time  $t$ , the troop needs to choose one action among several possible actions, which have a different impact on future movements. and those of cases give the different impacts. We assume that probability value of choosing ‘my action’ among ‘my possible action set’ during time  $t$  is same. That is, there’s no pre-defined preferred action to find my action. ‘My action’ is selected among ‘my possible action set’ during time  $t$ , which has minimum expected values when the action is occurred. The expected values are calculated by using the transition probability values that we discussed above.

For example, we have total 15 combinations of enemies’ location changing and 6 ‘my possible actions’ during time  $t$ . Using this information, we can find 6 expected costs at the end of time  $t$  by Bellman equation.

$$\begin{aligned}
 E\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) &= \left\{ A\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \end{bmatrix}\right) + \gamma \sum_{s' \in S} \text{prob}\{s'|s, x\} * \{E(s') + \mathcal{V}(s')\} \right\} = 18.4 \\
 E\left(\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) &= \left\{ A\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 2 \end{bmatrix}\right) + \gamma \sum_{s' \in S} \text{prob}\{s'|s, x\} * \{E(s') + \mathcal{V}(s')\} \right\} = 12.3 \\
 E\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}\right) &= \left\{ A\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 1 \end{bmatrix}\right) + \gamma \sum_{s' \in S} \text{prob}\{s'|s, x\} * \{E(s') + \mathcal{V}(s')\} \right\} = 15.7 \\
 E\left(\begin{bmatrix} 2 \\ 2 \end{bmatrix}\right) &= \left\{ A\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 2 \\ 2 \end{bmatrix}\right) + \gamma \sum_{s' \in S} \text{prob}\{s'|s, x\} * \{E(s') + \mathcal{V}(s')\} \right\} = 6.3 \\
 E\left(\begin{bmatrix} 3 \\ 1 \end{bmatrix}\right) &= \left\{ A\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 1 \end{bmatrix}\right) + \gamma \sum_{s' \in S} \text{prob}\{s'|s, x\} * \{E(s') + \mathcal{V}(s')\} \right\} = 9.5 \\
 E\left(\begin{bmatrix} 3 \\ 2 \end{bmatrix}\right) &= \left\{ A\left(\begin{bmatrix} 2 \\ 1 \end{bmatrix}, \begin{bmatrix} 3 \\ 2 \end{bmatrix}\right) + \gamma \sum_{s' \in S} \text{prob}\{s'|s, x\} * \{E(s') + \mathcal{V}(s')\} \right\} = 8.6
 \end{aligned}$$



 My optimal action during time t

[Expected cost for each action]

(1,1): 18.4

(1,2): 12.3

(2,1): 15.7

**(2,2): 6.3**

(3,1): 9.5

(3,2): 8.6

When a troop move from current position (2,1) to (2,2) during time t, its expected value is minimum among 6 actions.

**→ Optimal action during time t: (2,2)**

Figure 36. Enemies' location changing and their transition probability values

## 3.5 Solution approach and Approximations

The most significant problem of optimizing DP model is the exponentially increasing the computation time when the state size is growing. Moreover, as the number of hostile entities increases, the state size increases exponentially as we need to consider all the possible combinations. This results in long computation time to calculate optimal routing policies.

In this section, we use traditional value iteration approach and propose 2 novel approximation approaches to reduce the computation time required by value iteration. The approximation strategies are based on simulation.

### 3.5.1 Value iteration

Step 0. Initialization

Step 0a. Initialize  $V^0$  and  $S^0$ , where  $V$  is cost and  $S$  is state

Step 0b. Initialize  $t$  to be 1.

Step 1. While  $t \leq T$  (*maximum number of Iteration*)

Step 1a. For each state, solve the value function

Step 1b. Store  $V$  values and their optimal actions

Step 1c. Set  $t$  to  $t + 1$

Step 2. Return  $V$  values and their Optimal Actions

### 3.5.2 Decomposed Value Iteration (DVI)

#### Basic algorithm

Step 0. Solve the individual problems

Step 0a. Initialization

- Initialize  $V_i^0, S_i^0$ , where  $i$  is hostile entity,  $V$  is cost, and  $S$  is state
- Set  $t=0$

Step 0b. Iteration  $t$  times: Solve the model using value iteration for each enemy

Step 0c. Store  $V_i$  and  $a_i$

- The result of  $V_i$  and  $a_i$  are

$$V_i = \begin{bmatrix} V_{i,S_{i,1}} \\ \vdots \end{bmatrix}$$

$$a_i = [a_{i,S_{i,1}} \quad \dots]$$

$V_{i,S_{i,1}}$ :  $V$  value against to the hostile entity  $i$ , when the state is  $S_{i,1}$

$a_{i,S_{i,1}}$ : Optimal action against to the hostile entity  $i$ , when the state is  $S_{i,1}$

Step 1. Run simulations (iteration alpha times)

Step 1a. Initialization

- Initialize the state
- Store initialized state to “nextstate”, my initial location to “mynextloc”, and optimal action and  $V$  value for each enemy at initial state to “oai”.

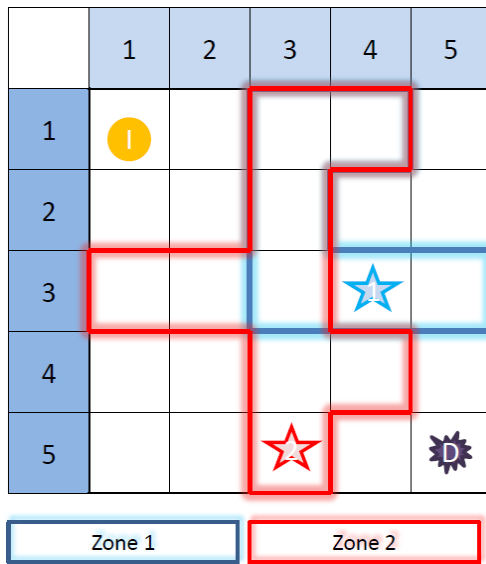
Step 1b. Find a path (till destination points)

- Choose the action that has minimum  $V$  value.
- Generate new “nextstate” by using random function.
- Calculate actual action cost and encounter cost.

Step 1c. If my location is destination point, then go to Step1a.

- Show transition from origin to destination
- Show cost information

**Example**



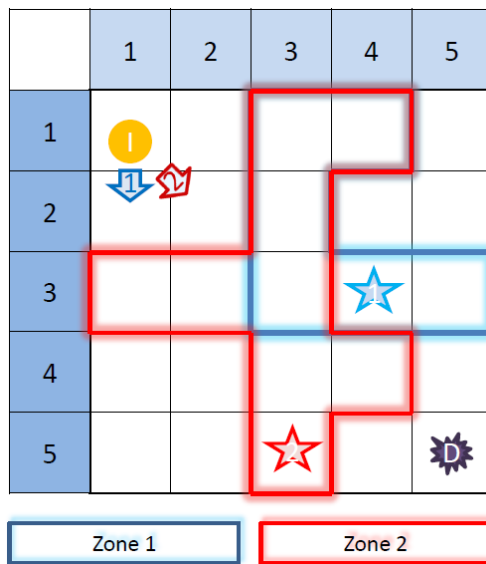
Initialize: Starting point

a.  $nextstate = \begin{bmatrix} (1,1) \\ (3,4) \\ (5,3) \end{bmatrix}$

b.  $mynextloc = (1,1)$

c.  $oa1 = (2,1), oa2 = (2,2)$

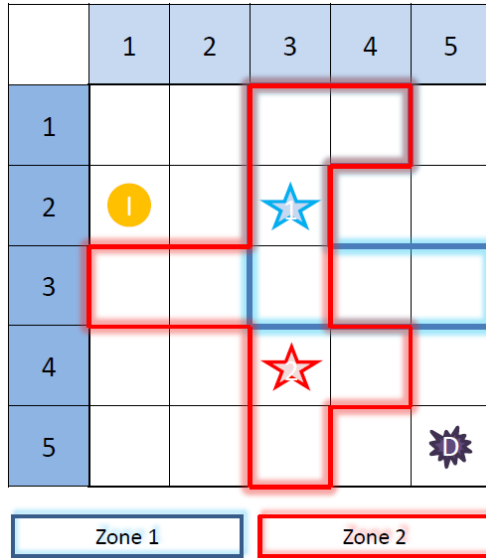
$V_1 \begin{pmatrix} (1,1) \\ (3,4) \end{pmatrix} = 10, V_2 \begin{pmatrix} (1,1) \\ (5,3) \end{pmatrix} = 11$



Choose the action that has minimum V

$V_1 \begin{pmatrix} (1,1) \\ (3,4) \end{pmatrix} = 10$

$V_2 \begin{pmatrix} (1,1) \\ (5,3) \end{pmatrix} = 11$



Generate nextstate

a.  $\text{nextstate} = \begin{bmatrix} (2,1) \\ (2,3) \\ (4,3) \end{bmatrix}$

b.  $\text{mynextloc} = (2,1)$

c.  $\text{oa1} = (3,2), \text{oa2} = (3,1)$

$V_1 \begin{pmatrix} (2,1) \\ (2,3) \end{pmatrix} = 9, V_2 \begin{pmatrix} (1,1) \\ (5,3) \end{pmatrix} = 10$

Figure 37. DVI approach

### 3.5.3 Modified Decomposed Value Iteration

#### Overview

The difference between 1<sup>st</sup> and 2<sup>nd</sup> approximation is choosing action at a certain state. In the 1<sup>st</sup> approximation we choose the action that has minimum V value among the individual cases. The problem of this method is we ignore the initial cost function and combinations of enemies' actions when we choose the next action. So, in the 2<sup>nd</sup> approximation, we deal with this problem using the modified cost function.

#### Basic algorithm

Step 0. Solve the individual problems: same as 1<sup>st</sup> approximation

Step 0a. Initialization

- Initialize  $V_i^0, S_i^0$ , where  $i$  is hostile entity,  $V$  is cost, and  $S$  is state
- Set  $t=0$

Step 0b. Iteration  $t$  times: Solve the model using value iteration for each enemy

Step 0c. Store  $V_i$  and  $a_i$

- The result of  $V_i$  and  $a_i$  are

$$V_i = \begin{bmatrix} V_{i,S_{i,1}} \\ \vdots \end{bmatrix}$$

$$a_i = [a_{i,S_{i,1}} \quad \dots]$$

$V_{i,S_{i,1}}$ : V value against to the hostile entity  $i$ , when the state is  $S_{i,1}$

$a_{i,S_{i,1}}$ : Optimal action against to the hostile entity  $i$ , when the state is  $S_{i,1}$



Step 1. Run simulations (iteration alpha times)

Step 1a. Initialization

- Initialize the
- Store initialized state to “nextstate”, my initial location to “mynextloc”, and optimal action and V value for each enemy at initial state to “oai”.

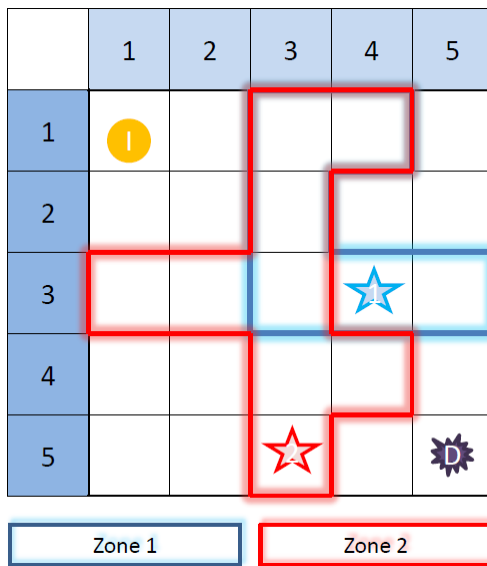
Step 1b. Find a path (till destination point)

- Calculate cost function using individual V value and optimal actions.
- Choose the action that has minimum cost
- Generate new “nextstate” by using random function.
- Calculate actual action cost and encounter cost.

Step 1c. If my location is destination point, then go to Step1a.

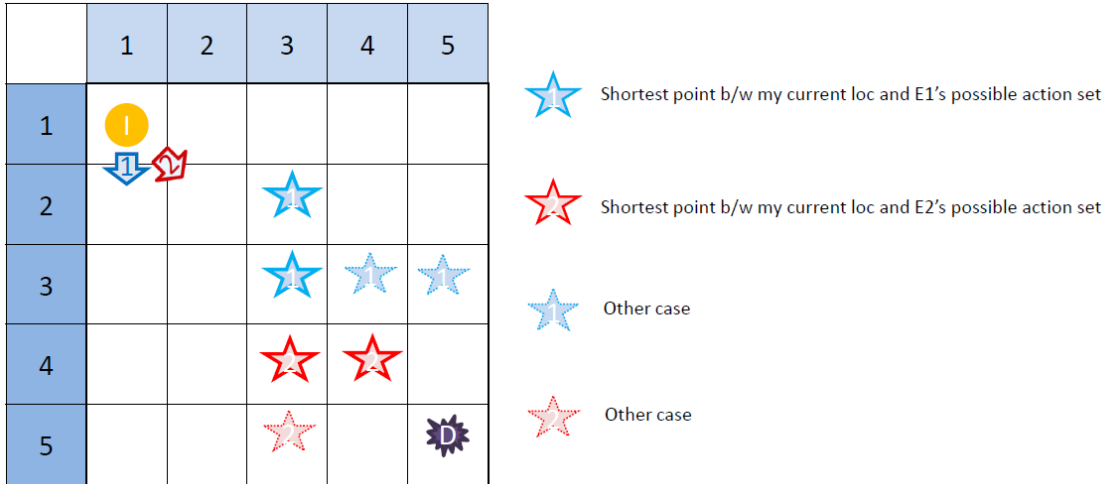
- Show transition from origin to destination
- Show cost information

Example



Initialize: Starting point

- a.  $\text{nextstate} = \begin{bmatrix} (1,1) \\ (3,4) \\ (5,3) \end{bmatrix}$
- b.  $\text{mynextloc} = (1,1)$
- c.  $\text{oa1} = (2,1), \text{oa2} = (2,2)$



a. Find Enemies' movement during time t

- 12 possible combinations

$$\begin{aligned} & \left[ \begin{matrix} (2,3) \\ (4,3) \end{matrix} \right] \text{ with prob } 0.12, \left[ \begin{matrix} (2,3) \\ (4,4) \end{matrix} \right] \text{ with prob } 0.12, \left[ \begin{matrix} (2,3) \\ (5,3) \end{matrix} \right] \text{ with prob } 0.08, \\ & \left[ \begin{matrix} (3,3) \\ (4,3) \end{matrix} \right] \text{ with prob } 0.12, \left[ \begin{matrix} (3,3) \\ (4,4) \end{matrix} \right] \text{ with prob } 0.12, \left[ \begin{matrix} (3,3) \\ (5,3) \end{matrix} \right] \text{ with prob } 0.08, \\ & \left[ \begin{matrix} (3,4) \\ (4,3) \end{matrix} \right] \text{ with prob } 0.07, \left[ \begin{matrix} (3,4) \\ (4,4) \end{matrix} \right] \text{ with prob } 0.07, \left[ \begin{matrix} (3,4) \\ (5,3) \end{matrix} \right] \text{ with prob } 0.04, \\ & \left[ \begin{matrix} (3,5) \\ (4,3) \end{matrix} \right] \text{ with prob } 0.07, \left[ \begin{matrix} (3,5) \\ (4,4) \end{matrix} \right] \text{ with prob } 0.07, \left[ \begin{matrix} (3,5) \\ (5,3) \end{matrix} \right] \text{ with prob } 0.04 \end{aligned}$$

b. Define new V value for Cost function

$$\begin{aligned} V \begin{pmatrix} 2 & 2 & 4 \\ 1 & 3 & 2 \end{pmatrix} &= V_1 \begin{pmatrix} 2 & 2 \\ 1 & 3 \end{pmatrix} + V_2 \begin{pmatrix} 2 & 4 \\ 1 & 2 \end{pmatrix} = 12 + 10 = 22 \\ &\vdots \\ V \begin{pmatrix} 2 & 2 & 4 \\ 2 & 3 & 2 \end{pmatrix} &= V_1 \begin{pmatrix} 2 & 2 \\ 2 & 3 \end{pmatrix} + V_2 \begin{pmatrix} 2 & 4 \\ 2 & 2 \end{pmatrix} = 11 + 13 = 24 \\ &\vdots \end{aligned}$$

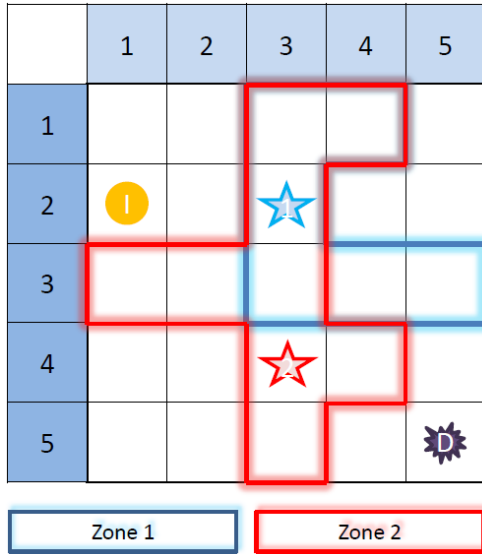
c. Calculate Cost function and choose the next position of 'I'

$$E(\text{move to } (2,1)) = \text{moving cost} + \sum_{i=1}^{12} \text{Prob} * (\text{encounter cost} + V) = 20,$$

where i are combination of enemies' movement during time t

$$E(\text{move to } (2,2)) = \text{moving cost} + \sum_{i=1}^{12} \text{Prob} * (\text{encounter cost} + V) = 23,$$

where i are combination of enemies' movement during time t



Generate nextstate

- a.  $\text{nextstate} = \begin{bmatrix} (2,1) \\ (2,3) \\ (4,3) \end{bmatrix}$
- b.  $\text{mynextloc} = (2,1)$
- c.  $\text{oa1} = (3,2), \text{oa2} = (3,1)$

Figure 38. Modified DVI approach

## 3.6 Numerical Experiment

### Experimental design

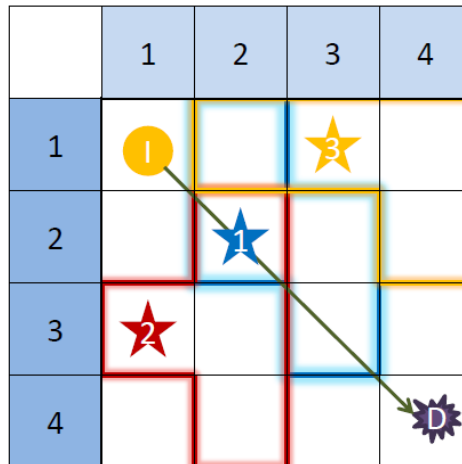


Figure 39. Map for experiment (initial state)

- Each party can predict the opponents' possible action during time  $t$ . That is, the troop has information of the enemies' territories at the beginning stage.
- Object: move origin to destination with minimum cost
  - Cost includes action cost and encounter cost
- Condition
  - 4 by 4 grid
  - Initial State (for simulation):  $S_0 = \begin{bmatrix} 1 & 2 & 3 & 1 \\ 1 & 2 & 1 & 3 \end{bmatrix}$
  - Action cost
    - Movement cost: 1
    - Staying cost: 2
  - 3 enemies: Each enemy has its own boundary.

- Encounter cost vector:  $\begin{bmatrix} 10 \\ 15 \\ 20 \end{bmatrix}$
- Discount rate: 1
- Enemy aggression vector:  $\begin{bmatrix} 1.6 \\ 1.8 \\ 1.4 \end{bmatrix}$

## Results

- Value iteration
  - Simulation result
    - Numbers of visiting at each cell

	1	2	3	4
1	20	76	0	0
2	1027	333	321	0
3	0	591	88	321
4	0	0	591	1000

Enemy 1	Enemy 2	Enemy 3
E1+E2	E1+E3	

The number at each cell means the number of visiting after the simulation. For example, 1027 at (2,1) is that a troop visited total 1027 during 1000 iterations.

20 at (1,1) means that a troop goes back to origin point from (2,1) because the enemies' positions are follows: enemy 1 at (4,2), enemy 2 at (3,3) and enemy 3 at (1,3).

The shaded cells are enemies' territories.

Figure 40. Value iteration result

- Numerical results

After the simulation, we get average 0.286 encounters and the average encounter cost of 3.565 per iteration and this means that a troop can meet enemies from origin to destination. Average step size (4.368) and average action cost (4.383) are similar, and this means that a troop rarely stays at a certain cell from origin to destination during the action.

- Suggested route

Usually, result shows that a troop keeps away from enemy 3 because of its high encounter cost. Sometimes a troop can pick the route such as  $\begin{bmatrix} 1 & 2 & 2 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 4 & 4 \end{bmatrix}$ , and in those cases, it can meet enemy 1 twice but it doesn't meet enemy 3 at the same time.

Based on the simulation result, the most frequent visiting route is  $\begin{bmatrix} 1 & 2 & 3 & 4 & 4 \\ 1 & 1 & 2 & 3 & 4 \end{bmatrix}$ , and when a troop takes this route, it can be possible to meet 2<sup>nd</sup> enemy during the action.

Another option can be  $\begin{bmatrix} 1 & 2 & 2 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 4 & 4 \end{bmatrix}$ , and when a troop takes this route, it can be possible to meet 1<sup>st</sup> enemy once or twice during the action.

- Decomposed value iteration

- Approximation result

- Value iteration results of single enemy cases

	1	2	3	4
1	4	4	3	3
2	3	2/3/3/3	2	2
3	3	2	1	1
4	3	2	1	0

E1: V value

	1	2	3	4
1	(2,1)	(1,3)	(2,4)	(2,4)
2	(3,3)	(3,3)/ (3,2)	(3,4)	(3,4)
3	(4,2)	(4,3)	(4,4)	(4,4)
4	(4,2)	(4,3)	(4,4)	

E1: Optimal Action

	1	2	3	4
1	4/4/4/3	3	3	3
2	4/4/4/3	2	2	2
3	3/4.792 /4.792/ 3	2	1	1
4	3/6.125 /6.125/ 6.947	2	1	0

E2: V value

	1	2	3	4
1	(1,2)/ (2,2)	(2,3)	(2,4)/ (2,3)	(2,4)
2	(1,2)/ (2,2)	(3,3)	(3,4)	(3,4)
3	(4,2)/ (2,1)/ (2,2)	(4,3)	(4,4)	(4,4)
4	(4,2)/ (3,2)	(4,3)	(4,4)	

E2: Optimal Action

	1	2	3	4
1	3	3	3	3
2	3	2	2	2
3	3	2	1	1
4	3	2	1	0

E3: V value

	1	2	3	4
1	(2,2)	(2,3)	(2,3)	(2,4)/ (2,3)
2	(3,2)	(3,3)	(3,4)	(3,4)
3	(3,2)	(4,3)	(4,4)	(4,4)
4	(4,2)	(4,3)	(4,4)	

E3: Optimal Action

Figure 41. Value iteration results of single enemy cases

- Numbers of visiting at each cell

	1	2	3	4
1	0	0	0	0
2	0	1000	0	0
3	0	0	1000	0
4	0	0	0	1000

Enemy 1	Enemy 2	Enemy 3
E 1 + E 2	E 1 + E 3	

When we pick the action which has minimum V value among 3 enemies, it follows the route that 3<sup>rd</sup> enemy's optimal action. It is because the 3<sup>rd</sup> enemy's v value of each cell, (1,1), (2,2), (3,3) and (4,4), is the smallest among 3 cases.

The shaded cells are enemies' territories.

Figure 42. DVI result

- Numerical results

After applying the 1<sup>st</sup> algorithm, we get average 0.657 encounters and average encounter cost of 7.63 per iteration and this means that a troop can meet enemies from origin to destination. Average step size and average action cost are 3, and this means that a troop doesn't stay at a certain cell.

- Suggested route

Based on the result, there's only one route,  $\begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 3 & 4 \end{bmatrix}$ , and this is quite different from value iteration result. However, it gives the same pattern, which is a troop avoids enemy 3 during the action.



- Modified decomposed value iteration
  - Approximation result
    - Numbers of visiting at each cell

	1	2	3	4
1	0	127	0	0
2	1000	394	127	0
3	0	556	317	127
4	0	0	556	1000

Enemy 1	Enemy 2	Enemy 3
E 1 + E 2		E 1 + E 3

When we pick the action which has minimum expected value among 3 actions given by single enemy case, the result is quite similar to the value iteration result. The difference from 1<sup>st</sup> approximation is that 2<sup>nd</sup> approximation considers the combination of enemies' movements during the simulation.

The shaded cells are enemies' territories.

Figure 43. modified DVI result

- Numbers of visiting at each cell

After applying 2<sup>nd</sup> algorithm, we get average 0.327 encounters and average encounter cost of 3.85 per iteration and this means that a troop can meet enemies from origin to destination. Average step size and average action cost are 4.204, and this means that a troop doesn't stay at a certain cell.

- Suggested route

Usually, result shows that a troop avoids from enemy 3 during the action.

Based on the result, most frequent visiting route is  $\begin{bmatrix} 1 & 2 & 3 & 4 & 4 \\ 1 & 1 & 2 & 3 & 4 \end{bmatrix}$ , and when a troop takes this route, it can meet enemy 2 during the action.

The other options can be  $\begin{bmatrix} 1 & 2 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 4 \end{bmatrix}$  or  $\begin{bmatrix} 1 & 2 & 2 & 3 & 4 & 4 \\ 1 & 1 & 2 & 2 & 3 & 4 \end{bmatrix}$ , and when a troop picks one route between both routes, it can meet enemy 1, enemy 2 or both enemies during the action.

When a troop picks the route  $\begin{bmatrix} 1 & 2 & 1 & 2 & 3 & 4 \\ 1 & 1 & 2 & 3 & 4 & 4 \end{bmatrix}$ , it can meet enemy 1, enemy 3 or both enemies during the action.

- Result Analysis

From the all the solutions from the experiments, we can get the similar pattern which is the suggested route has a tendency that is avoiding enemy 3 during the action. Moreover, 2<sup>nd</sup> approximation gave more similar pattern to the value iteration result because we consider the actual cost function and combinations of enemies' movements in the algorithm. Table 19 shows the brief results of the experiment.

	Computation time(min)	avg. Encounts	avg. step size	avg. total cost
Value Iteration	117	0.286	4.368	7.948
DVI	1	0.657	3	10.63
m. DVI	14	0.327	4.204	8.054

Table 44. Brief results of experiment

- Computation time (CPU: Intel Core 2 Duo 1.2Ghz, Ram: 2GB)

Figure 36 shows the total computational time and it shows that both approximation methods decrease the computational time significantly.

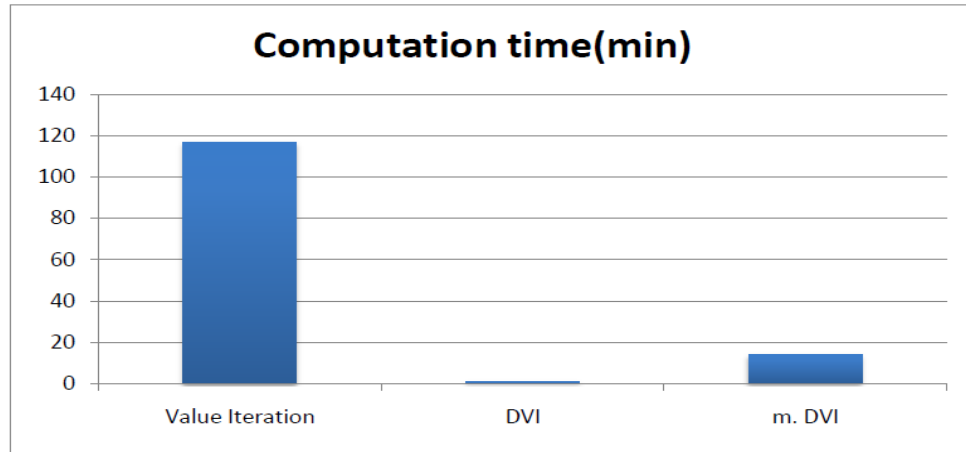


Figure 44. Total computation time result

- Average number of encounters during the action

Figure 37 shows the average number of encounters during the action. Compared to the value iteration result, approximation results generally give more encounters during the action; however, modified DVI result is quite similar to the value iteration result.

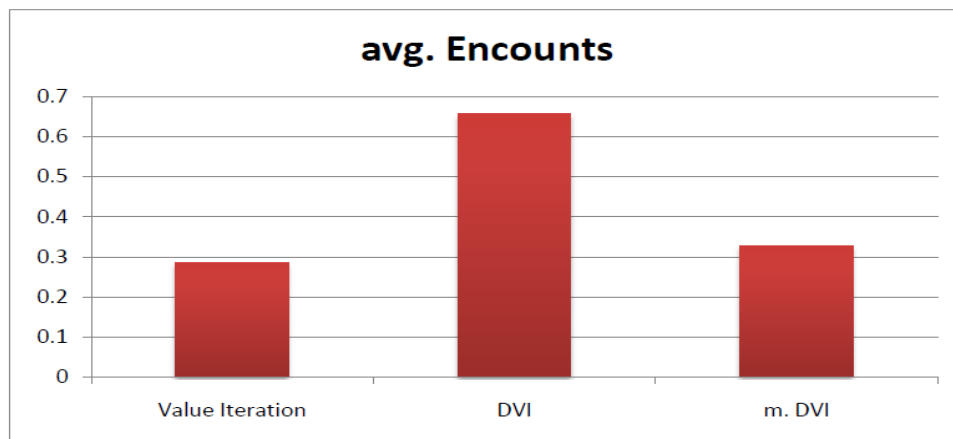


Figure 45. Average number of encounters

- Average step sizes during the action

Figure 38 shows the average step sizes during the action. Compared to the value iteration, modified DVI gives better result than DVI.

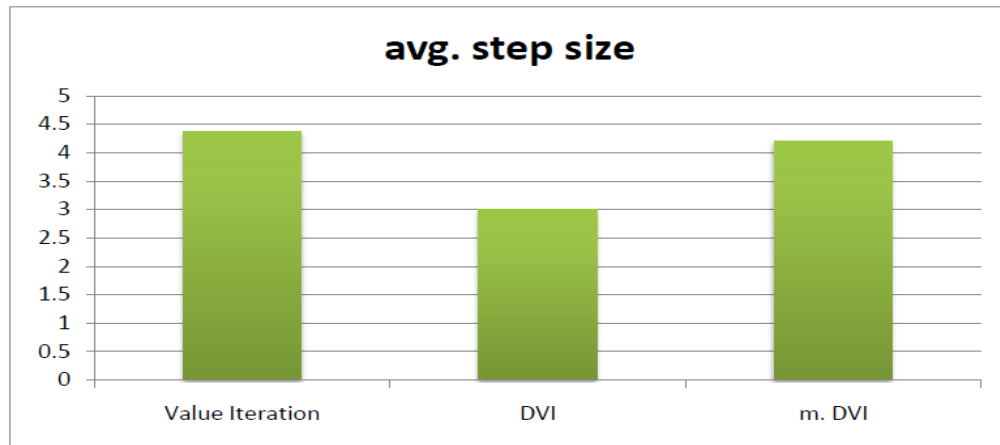


Figure 46. Average number of step sizes

- Average total cost during the action

Figure 39 shows the average total cost during the action. Compared to the value iteration, modified DVI gives better result than DVI because DVI gives more encounters during the action.

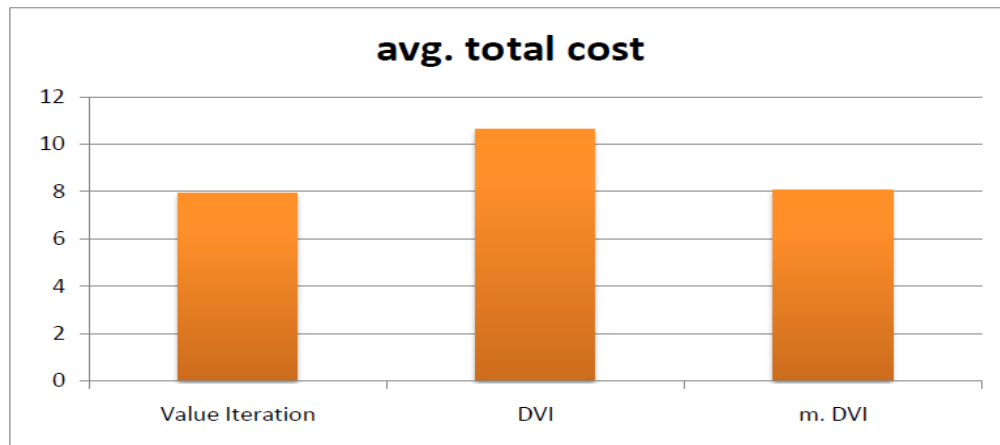


Figure 47. Average total cost

### 3.7 Conclusion

In this chapter, we have discussed about the multi-objective routing problem with Markov decision process. We have considered traveling distance and risk (or encounters) in the model. To solve the problem, we have applied 3 types of algorithms: Value iteration, approximation with individual value iteration, and approximation with individual value iteration and cost function. Although the value iteration gives more accurate solutions, it needs lots of computational resources and time consuming. The other 2 approximation techniques solve this computational burden with manageable solutions and 2<sup>nd</sup> approximation which considers cost function and individual value iterations gives quite similar to the traditional value iteration approach.

# CHAPTER4. CONCLUSION

---

In this thesis, we have dealt with 2 different types of model to solve the military routing problem. The 1<sup>st</sup> model is multi-objective integer programming which considers traveling distance, traveling risk, and maximum flow of given path. The 2<sup>nd</sup> model is Markov decision process which considers traveling distance and traveling risk (or enemies' encounters) during the action.

Each approach has distinctive properties:

1. Multi-objective integer programming: In this approach, we can get the solutions relatively faster than Markov decision model and gives various solution sets depends on the priorities. However, we cannot adapt the dynamic nature of military operations.
2. Markov decision process: The most important advantage of this approach is that we can adapt the military environment changings dynamically. However, it needs relatively long computation time to make a military decision.

In this thesis, both of 2 models develop the various solution approaches, so decision maker can choose the appropriate technique. We use the 3 types of solution approach for multi-objective integer programming: weighted sum approach, goal programming, and lexicographical goal programming. And to get the solution sets for Markov decision process, we use the traditional value iteration, approximation with individual value iteration which means pruning the combinations of all the possible information, and approximation with individual value iteration and cost function.

The result of each model gives the various options to the decision maker; however, none of them are optimal solution set. Since there can be the different conditions in each military operation, decision maker needs to consider all the factors when he/she choose the solution approach or solution set.

## REFERENCES

1. A.Ieland and M-J Oboroceanu, 2010, American War and Military Operations Casualties: Lists and Statistics, CRS report for Congress
2. A.Stepanove and J-M Smith, 2009 Multi-objective evacuation routing in transportation networks, European Journal of Operational Research, Vol. 198, pp.435-446
3. M.Ozlen and M.Azizoglu, 2009, Multi-objective integer programming: A general approach for generating all non-dominated solutions, European Journal of Operational Research, Vol. 199, pp.25-35
4. Murray-Tuite P.M., 2008, Transportation Network Risk Profile for an Origin-Destination Pair, Transportation Research Record: Journal of the Transportation Research Board, No. 2041, pp. 19-28
5. Murray-Tuite P.M., and Mahmassani H.S., 2004, Methodology for Determining Vulnerable Links in a Transportation Network, Transportation Research Record: Journal of the Transportation Research Board, No. 1882,pp. 88-96
6. Murray-Tuite, P., 2005, Methodology for Determining Transportation Network Connectivity Reliability Under Threats of Terrorism, Proc., International SIV Congress on People, Land, Environment, and Transport Infrastructures, Bari, Italy
7. Sun C., Ritchie S.G., Tsai K., and Jayakrishnan R., 1999, Use of vehicle signature for vehicle reidentification on freeways, Transportation Research Part C 7, pp. 167-185
8. Wadhwa V., and Ravindarn A.R. 2007, Vendor selection in outsourcing, Computer & Operations Research, Vol. 34, Issue 12, PP. 3725-3737
9. Ford L. R. Jr., and Fulkerson D.R., 1956, MAXIMAL FLOW THROUGH A NETWORK, Canadian Journal of Mathematics, PP. 399-404
10. Ford L. R. Jr., and Fulkerson D.R., 1962, Flows in Networks, Princeton University Press, Princeton, N.J., 1962
11. Ignizio J.P., 1978, A Review of Goal Programming: A Tool for Multiobjective Analysis, The Journal of the operations Research Society, Vol. 29, No. 11, pp. 1109-1119
12. Carlyle W.M., Royset J.O., and Wood R.K., 2009, Routing military aircraft with a constrained shortest-path algorithm, Military Operations Research
13. Akgun V., Parekh A., and Batta R., 2007, Routing of a hazmat truck in the presence of weather systems, Computers & Operations Research, Vol. 34, Issue 5, pp. 1351-1373



14. Zografos K.G., and Androutsopoulos K.N., 2004, A heuristic algorithm for solving hazardous materials distribution problems, *European Journal of Operations Research*, Vol. 152, Issue 2, pp. 507-519
15. Fourer R., Gay D.M., and Kernighan B.W., 2002, *AMPL: A Modeling Language for Mathematical Programming*, second edition, Duxbury Press/ Brooks/ Cole Publishing Company
16. iCasualties: Operation Iraqi freedom and Operation Enduring Freedom Casualties, <http://icasualties.org/Iraq/Fatalities.aspx> (accessed 06-22-10)
17. Schniederjans M. J., 1995, *Goal Programming, Methodology and Applications*, Springer-Verlag New York, LLC
18. Romero C., 2001, Extended lexicographic goal programming: a unifying approach, *OMEGA: The International Journal of Management Science* 29, pp 63-71
19. Aberdeen D., Thiebaut S., and Zhang L., 2004, Decision-Theoretic Military Operation Planning, In *Proc. of ICPAS'04*, Vol. 14, pp. 402-411, AAAI, June 2004
20. Chang H.S. and Marcus S., 2003, Approximate receding horizon approach for Markov decision processes: average reward case, *Journal of Mathematical Analysis and Application*, 286, pp. 636-651
21. Derman C., 1962, ON SEQUENTIAL DECISION AND MARKOV CHAINS, *Management Science*, Vol.9, No.1, pp.16-24
22. Garcia-Hernandez M.G., Ruiz-Pinales J., Reyes-Ballesteros A., Onaindia E., Avina-Cervantes J.G., Ledesma S., and Hernandez D, *Association Rule-Based Markov Decision Processes*
23. Meirina C., Levchuk Y.N., Levchuk C.M., and Pattipati K.R., 2005, *Goal Management in Organizations: A Markov Decision Process (MDP) Approach*
24. Smith T., and Simmons R., 2006, Focused Real-Time Dynamic Programming for MDPs: Squeezing More out of a Heuristic, In *Proc. Nat. Conf. on Artificial Intelligence (AAAI)*
25. Psaraftis H.N., 1995, Dynamic vehicle routing: Status and prospects, *Annals of Operations Research*, Vol.61, No.1, pp.143-164
26. Hayes M. and Norman J.M., 1984, Dynamic Programming in Orienteering: Route Choice and the Siting of Controls, *Journal of Operational Research Society*, Vol.35, No.9, pp.791-796

27. Secomandi N., A Rollout Policy for the Vehicle Routing Problem, 2001, Operations Research, Vol.49, No.5, pp.796-802
28. Dror M., Laporte G., and Trudeau P., 1989, Vehicle Routing with Stochastic Demands: Properties and Solution Frameworks, Transportation Science, Vol.23, No.3, pp.166-176
29. Pineau J., Gordon G., and Thrun S., 2003, Point-based value iteration: An anytime algorithm for POMDPs, In Proc. Int. Joint Conf. on Artificial Intelligence, Acapulco, Mexico
30. Littman M.L., Dean T.L., and Kaelbling L.P., 1995, On the complexity of solving Markov decision problems, In Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence, pp.394-401, Montreal, Canada
31. Smallwood R.D., 1973, The Optimal Control of Partially Observable Markov Processes Over a Finite Horizon, Operations Research, Vol.21, No.5, pp.1071-1088
32. Powell W.B., 2007, Approximate Dynamic Programming: Solving the Curses of Dimensionality, 1<sup>st</sup> edition, Wiley-Interscience
33. Winston, W.L., 2003, Introduction to Probability Models: Operations Research, Volume II, 4<sup>th</sup> edition, Duxbury Press

# APPENDIX

## 1. AMPL Code

### 1.1. Weighted sum approach

#### 1.1.1. Mod file

```
# This mod file consider three objective case
# first one is minimizing total distance,
# second one is minimizing total risk,
# and last one is min-max problem of first objectives (max capacity)

#*****sets*****

set N; # set of every node (B+I+T)
set B within N; # set of military base
set I within N; # set of intermediary transportation nodes
set T within N; # set of military target
set A within (N cross N); # set of links between node i and node j

set M; # set of types of military troop

#*****parameters*****

param dist{A}>=0; # distance of path (i,j) in A
param risk{A}>=0; # risk of path (i,j) in A

param a{B,M}>=0; # availability of military type (M) at the base(B)
param r{T,M}>=0; # requirement of military type (M) at the target(T)

param alpha; # 1st weight of the objective
param beta; # 2nd weight of the objective

param limit{A} >=0; # maximum path limitation. In this model, I didn't consider this.

# optimal value of total distance when we only consider the single objective - minimizing total distance
param ld;

# optimal value of total risk when we only consider the single objective - minimizing total risk
param lr;

# optimal value of max capacity
param lm;

#*****Variables*****

var x{(i,j) in A, k in M} >= 0 integer; # number of troops from node i to j
var MM{k in M}; # maximum flow
```

```

*****objective function*****

minimize multi_objectives:

((alpha-1)/10)*(1/d)*sum{(i,j) in A, k in M} dist[i,j]*x[i,j,k]
+ ((beta-1)/10)*(1/lm)*sum{k in M} MM[k]
+ (1-((alpha-1)+(beta-1))/10)*(1/lr)*sum{(i,j) in A, k in M} risk[i,j]*x[i,j,k];

*****Constraints*****

# definition of M
subject to MM_def{i in N, k in M}: MM[k] >= sum{(i,j) in A} dist[i,j]*x[i,j,k];

# availability of base i
subject to availability{b in B, m in M}: sum{(b,j) in A} x[b,j,m] <= a[b,m];

#flow balance
subject to flowbalance{l in I, m in M}: sum{(i,l) in A} x[i,l,m] = sum{(l,j) in A} x[l,j,m];

# required elements of target j
subject to requirements{f in T, m in M}: sum{(i,f) in A} x[i,f,m]>=r[f,m];

```

### 1.1.2. Dat file

```

data;

***** define sets and their elements *****

set B:=1,2,3; # base
set I:=A,B,C,D,E,F,G,H,I,J,K,L,M; # intermediary nodes
set T:=T1,T2,T3,T4; # targets
set N:=1,2,3,A,B,C,D,E,F,G,H,I,J,K,L,M,T1,T2,T3,T4; # entire nodes in the system

set M:=tk,bb,ms; # types of elements

set A:= (1,A) (1,B) (2,B) (2,C) (2,E) (3,C) (3,E) # paths
(A,B) (B,A) (A,D) (D,A) (A,F) (F,A) (A,L) (L,A)
(B,C) (C,B) (B,D) (D,B) (B,E) (E,B)
(C,E) (E,C) (C,H) (H,C) (C,T4)
(D,G) (G,D) (D,I) (I,D)
(E,G) (G,E)
(F,G) (G,F) (F,L) (L,F)
(G,H) (H,G) (G,J) (J,G) (G,K) (K,G)
(H,M) (M,H) (H,T4)
(I,J) (J,I) (I,L) (L,I) (J,T1) (J,T3)
(K,M) (M,K) (K,T3)
(L,T1) (L,T2)
(M,T2) (M,T4);

param alpha:=1;
param beta:=1;

```

```
# availabilities
param a:
    tk    bb    ms:=
1      25    10    5
2      20    25    30
3      5     15    10;
```

```
#requiements
param r:
    tk    bb    ms:=
T1     15    10    10
T2     10    15    10
T3     5     10    15
T4     10    5     5;
```

```
# distance information
param dist default 1000:=
    1,A    10.2
    1,B     9.9
    2,B     5.0
    2,C    13.4
    2,E    14.3
    3,C     9.2
    3,E    16.5
    A,B     9.5
    B,A     9.5
    A,D     7.6
    D,A     7.6
    A,F     9.5
    F,A     9.5
    A,L    23.0
    L,A    23.0
    B,C    12.0
    C,B    12.0
    B,D     6.3
    D,B     6.3
    B,E    10.0
    E,B    10.0
    C,E     9.2
    E,C     9.2
    C,H    17.1
    H,C    17.1
    C,T4   39.1
    D,G     8.9
    G,D     8.9
    D,I    14.6
    I,D    14.6
    E,G     7.2
    G,E     7.2
    F,G     8.2
    G,F     8.2
    F,L    14.1
    L,F    14.1
    G,H     8.9
    H,G     8.9
```

G,J	7.6
J,G	7.6
G,K	10.4
K,G	10.4
H,M	12.2
M,H	12.2
H,T4	22.4
I,J	5.1
J,I	5.1
I,L	6.3
L,I	6.3
J,T1	15.8
J,T3	10.2
K,M	6.7
M,K	6.7
K,T3	8.1
L,T1	8.2
L,T2	13.0
M,T2	13.6
M,T4	11.7;

# risk information

param risk default 1000:=-

1,A	1.31
1,B	0
2,B	0
2,C	2.94
2,E	2.88
3,C	0
3,E	2.97
A,B	0
B,A	0
A,D	3.62
D,A	3.62
A,F	3.61
F,A	3.61
A,L	12.71
L,A	12.71
B,C	2.90
C,B	2.90
B,D	0
D,B	0
B,E	2.94
E,B	2.94
C,E	2.97
E,C	2.97
C,H	5.07
H,C	5.07
C,T4	5.15
D,G	4.04
G,D	4.04
D,I	3.93
I,D	3.93
E,G	3.39
G,E	3.39

F,G	0
G,F	0
F,L	9.00
L,F	9.00
G,H	0
H,G	0
G,J	0.99
J,G	0.99
G,K	2.47
K,G	2.47
H,M	6.62
M,H	6.62
H,T4	0
I,J	0
J,I	0
I,L	6.23
L,I	6.23
J,T1	6.29
J,T3	0
K,M	6.63
M,K	6.63
K,T3	1.34
L,T1	0
L,T2	0
M,T2	0
M,T4	0;

```
# lower bounds
param ld:=5135.5;
param lr:=437.1;
param lm:=915.9;
```

### 1.1.3. Run file

```
model weightedsum.mod;
data weightedsum.dat;
for {1..12}
{
  for {1..12-alpha}
  {
    solve;
    display (alpha-1)/10 >> weightedsum.txt;
    display (beta-1)/10 >> weightedsum.txt;
    display (12-alpha-beta)/10 >> weightedsum.txt;
    display multi_objectives >> weightedsum.txt;
    display sum{(i,j) in A, k in M} dist[i,j]*x[i,j,k] >> weightedsum.txt;
    display sum{k in M} MM[k] >> weightedsum.txt;
    display sum{(i,j) in A, k in M} risk[i,j]*x[i,j,k] >> weightedsum.txt;
    display x >> weightedsum.txt;
    display MM >> weightedsum.txt;
    let beta:=beta+1;
  }
  let alpha:=alpha+1;
  let beta:=1;
}
```

## 1.2. Goal programming approach: mod file

```

# This mod file consider three objective case
# first one is minimizing total distance,
# second one is minimizing total risk,
# and last one is min-max problem of first objectives (max capacity)

#*****sets*****

set N;                                # set of every node (B+I+T)
set B within N;                       # set of military base
set I within N;                       # set of intermediary transportation nodes
set T within N;                       # set of military target
set A within (N cross N);             # set of links between node i and node j

set M;                                # set of types of military troop

#*****parameters*****

param dist{A}>=0;                      # distance of arc (i,j) in A
param risk{A}>=0;                      # risk of arc (i,j) in A

param a{B,M}>=0;                      # availability of military type (M) at the military base(B)
param r{T,M}>=0;                      # requirement of military type (M) at the military target(T)

param alpha;                          # 1st weight of the objective
param beta;                            # 2nd weight of the objective

param limit{A} >=0;                   # maximum path limitation. In this model, I didn't consider this.

#*****Variables*****

var x{(i,j) in A, k in M} >= 0 integer; # number of troops from node i to j
var MM{k in M};                       # maximum flow

# goal difference variables
var d1p>=0 integer;
var d1m>=0 integer;
var d2p>=0 integer;
var d2m>=0 integer;
var d3p>=0 integer;
var d3m>=0 integer;

#*****objective function*****

minimize multi_goal: ((alpha-1)/10)*d1p + ((beta-1)/10)*d2p+ (1-((alpha-1)+(beta-1))/10)*d3p;

#*****Constraints*****

# definition of M
subject to MM_def{i in N, k in M}: MM[k] >= sum{(i,j) in A} dist[i,j]*x[i,j,k];

```



```

# availability of base i
subject to availability{b in B, m in M}: sum{(b,j) in A} x[b,j,m] <= a[b,m];

#flow balance
subject to flowbalance{l in I, m in M}: sum{(i,l) in A} x[i,l,m] = sum{(l,j) in A} x[l,j,m];

# required troops of military target j
subject to requirements{f in T, m in M}: sum{(i,f) in A} x[i,f,m]>=r[f,m];

# goal constraints
subject to mindistgoal: sum{(i,j) in A, k in M} dist[i,j]*x[i,j,k]+d1m-d1p=5649;
subject to minmaxflowgoal: sum{k in M} MM[k]+d2m-d2p=481;
subject to minriskgoal: sum{(i,j) in A, k in M} risk[i,j]*x[i,j,k]+d3m-d3p=1007;

```

### 1.3. Lexicographic goal programming approach: mod file

```

# This mod file consider three objective case
# first one is minimizing total distance,
# second one is minimizing total risk,
# and last one is min-max problem of first objectives (max capacity)

#*****sets*****

set N;                # set of every node (B+I+T)
set B within N;      # set of base
set I within N;      # set of intermediary transportation nodes
set T within N;      # set of target
set A within (N cross N);  # set of links between node i and node j

set M;                # set of types of military troop

#*****parameters*****

param dist{A}>=0;     # distance of arc (i,j) in A
param risk{A}>=0;     # risk of arc (i,j) in A

param a{B,M}>=0;      # availability of military type (M) at the military base(B)
param r{T,M}>=0;      # requirement of military type (M) at the military target(T)

param alpha;         # 1st weight of the objective
param beta;          # 2nd weight of the objective

param limit{A} >=0;  # maximum path limitation. In this model, I didn't consider this.

#*****Variables*****
*
var x{(i,j) in A, k in M} >= 0 integer;  # number of troops from node i to j
var MM{k in M};                          # maximum flow

```

```

*****object function*****
#1st priority: min total distance
# minimize total_distance: sum{(i,j) in A, k in M} dist[i,j]*x[i,j,k];

#2nd priority: min max flow
#minimize maximizing_flow: sum{k in M} MM[k];

#3rd priority: min total risk
minimize total_risk: sum{(i,j) in A, k in M} risk[i,j]*x[i,j,k];

*****Constraints*****

# definition of M
subject to MM_def{i in N, k in M}: MM[k] >= sum{(i,j) in A} dist[i,j]*x[i,j,k];

# availability of base i
subject to availability{b in B, m in M}: sum{(b,j) in A} x[b,j,m] <= a[b,m];

#flow balance
subject to flowbalance{l in I, m in M}: sum{(i,l) in A} x[i,l,m] = sum{(l,j) in A} x[l,j,m];

# required troops of military target j
subject to requirements{f in T, m in M}: sum{(i,f) in A} x[i,f,m]>=r[f,m];

#1st priority goal
#subject to first_priority_goal: sum{(i,j) in A, k in M} dist[i,j]*x[i,j,k] <= 5392;

#2nd priority goal
#subject to second_priority_goal: sum{k in M} MM[k] <= 499;

```

## 2. Experimental design: multi objective integer programming

### 2.1. Brief description of the map

Brief Description of the map														
X-COOR	Y-COOR	PAIR	X-DIF	Y-DIF	DISTANCE	RISK	SLOPE	b	P_SLOPE1	P_b1	Intersect_x1	Intersect_y1	dist_1	W/r
3	-4	1,A	-10	-2	10.2	1.31%	0.20	-4.6	-5.00	30.0	6.7	-3.3	1.77	0
3	-4	1,B	-7	7	9.9	0.00%	-1.00	-1.0	1.00	-12.0	5.5	-6.5	2.12	1
5	-11	2,B	-5	0	5.0	0.00%	0.00	-11.0				-11.0		
5	-11	2,C	-6	12	13.4	2.94%	-2.00	-1.0	0.50	-23.0	8.8	-18.6	3.58	0
5	-11	2,E	-13	6	14.3	2.88%	-0.46	-8.7	2.17	-43.0	13.1	-14.7	2.51	0
2	-21	3,C	-9	2	9.2	0.00%	-0.22	-20.6				-20.6		
2	-21	3,E	-16	-4	16.5	2.97%	0.25	-21.5	-4.00	31.0	12.4	-18.4	1.46	0
13	-2	A,B	3	9	9.5	0.00%	3.00	-41.0				-41.0		
13	-2	A,D	-3	7	7.6	3.62%	-2.33	28.3	0.43	-14.3	15.4	-7.7	1.71	0
13	-2	A,F	-9	3	9.5	3.61%	-0.33	2.3	3.00	-58.0	18.1	-3.7	3.48	0
13	-2	A,L	-23	1	23.0	12.71%	-0.0435	-1.4	23.00	-398.0	17.2	-2.2	4.82	0
		A,L	-23	1					23.00	-603.0	26.1	-2.6	2.43	0
		A,L	-23	1					23.00	-787.0	34.1	-2.9	2.08	0
10	-11	B,C	-1	12	12.0	2.90%	-12.00	109.0	0.08	-18.0	10.5	-17.1	1.49	0
10	-11	B,D	-6	-2	6.3	0.00%	0.33	-14.3				-14.3		
10	-11	B,E	-8	6	10.0	2.94%	-0.75	-3.5	1.33	-33.0	14.2	-14.1	3.60	0
11	-23	C,E	-7	-6	9.2	2.97%	0.86	-32.4	-1.17	-3.0	14.5	-20.0	3.90	0
11	-23	C,H	-17	-2	17.1	5.07%	0.12	-24.3	-8.50	175.5	23.2	-21.6	1.58	0
11	-23	C,T4	-39	2	39.1	5.15%	-0.05	-22.4	19.50	-468.5	22.8	-23.6	3.61	0
16	-9	D,G	-8	4	8.9	4.04%	-0.50	-1.0	2.00	-41.0	16.0	-9.0	2.24	0
		D,G	-8	4					2.00	-53.0	20.8	-11.4	1.79	0
16	-9	D,I	-14	-4	14.6	3.93%	0.29	-13.6	-3.50	52.5	17.5	-8.6	1.65	0
		D,I	-14	-4					-3.50	86.0	26.3	-6.1	1.10	0
18	-17	E,G	-6	-4	7.2	3.39%	0.67	-29.0	-1.50	17.0	21.2	-14.8	2.22	0
22	-5	F,G	-2	8	8.2	0.00%	-4.00	83.0	0.25	-11.3	22.2	-5.7	5.34	1
22	-5	F,L	-14	-2	14.1	9.00%	0.14	-8.1	-7.00	177.0	25.9	-4.4	0.6	0
		F,L	-14	-2					-7.00	233.0	33.8	-3.3	1.7	0
24	-13	G,H	-4	8	8.9	0.00%	-2.00	35.0	0.50	-31.5	26.6	-18.2	4.02	1
24	-13	G,J	-7	-3	7.6	0.99%	0.43	-23.3	-2.33	52.3	27.4	-11.6	1.58	0
24	-13	G,K	-10	3	10.4	2.47%	-0.30	-5.8	3.33	-106.3	27.7	-14.1	1.15	0
		G,K	-10	3					3.33	-131.3	34.6	-16.2	1.92	0
28	-21	H,M	-12	-2	12.2	6.62%	0.17	-25.7	-6.00	186.0	34.3	-19.9	1.97	0
28	-21	H,T4	-22	4	22.4	0.00%	-0.18	-15.9	5.50	-205.0	33.3	-22.0	4.02	1
30	-5	I,J	-1	5	5.1	0.00%	-5.00	145.0				145.0		
30	-5	I,L	-6	-2	6.3	6.23%	0.33	-15.0	-3.00	97.0	33.6	-3.8	1.26	0
31	-10	J,T1	-13	-9	15.8	6.29%	0.69	-31.5	-1.44	44.1	35.4	-7.0	2.40	0
31	-10	J,T3	-10	2	10.2	0.00%	-0.20	-3.8				-3.8		
34	-16	K,M	-6	3	6.7	6.63%	-0.50	1.0	2.00	-86.0	34.8	-16.4	1.79	0
34	-16	K,T3	-7	-4	8.1	1.34%	0.57	-35.4	-1.75	41.5	33.1	-16.5	1.74	0
36	-3	L,T1	-8	-2	8.2	0.00%	0.25	-12.0	-4.00	141.0	36.0	-3.0		
36	-3	L,T2	-12	5	13.0	0.00%	-0.42	12.0	2.40	-89.4	36.0	-3.0		
40	-19	M,T2	-8	-11	13.6	0.00%	1.38	-74.0				-74.0		
40	-19	M,T4	-10	6	11.7	0.00%	-0.60	5.0				5.0		

2.2. Coordinates of each node

INFO.	X-COOR	Y-COOR
1	3	-4
2	5	-11
3	2	-21
A	13	-2
B	10	-11
C	11	-23
D	16	-9
E	18	-17
F	22	-5
G	24	-13
H	28	-21
I	30	-5
J	31	-10
K	34	-16
L	36	-3
M	40	-19
T1	44	-1
T2	48	-8
T3	41	-12
T4	50	-25

### 2.3. Risk Zone

<b>zone 1</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.6	0.9	2	7	-5
Vehicle-Borne IED	20000	0.2				
Suicide Bumb IED	100	0.2				
<b>zone 2</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.5	0.95	4	12	-17
Vehicle-Borne IED	20000	0.3				
Suicide Bumb IED	100	0.2				
<b>zone 3</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.3	1.1	5	17	-7
Vehicle-Borne IED	20000	0.3				
Suicide Bumb IED	100	0.4				
<b>zone 4</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.4	1	3	20	-13
Vehicle-Borne IED	20000	0.4				
Suicide Bumb IED	100	0.2				
<b>zone 5</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.3	1.05	4	23	-20
Vehicle-Borne IED	20000	0.5				
Suicide Bumb IED	100	0.2				
<b>zone 6</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.2	1.1	3	26	-5
Vehicle-Borne IED	20000	0.3				
Suicide Bumb IED	100	0.5				
<b>zone 7</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.6	0.95	3	28	-13
Vehicle-Borne IED	20000	0.1				
Suicide Bumb IED	100	0.3				
<b>zone 8</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.2	1	4	34	-18
Vehicle-Borne IED	20000	0.7				
Suicide Bumb IED	100	0.1				
<b>zone 9</b>						
IED type	Destuctive Power (lb)	Probability	Ground	radius	central_x	central_y
package type IED	1000	0.1	1.1	3	34	-5
Vehicle-Borne IED	20000	0.7				
Suicide Bumb IED	100	0.2				

2.4. Distance between intersection point and arc

PATH	Int_x	Int_y	dist b/w S&I	dist b/w E&I
1,A	6.7	-3.3	3.765045216	0.234954784
2,C	8.8	-18.6	7.577708764	0.422291236
2,E	13.05366	-14.7171	1.485727945	6.514339057
3,E	12.35294	-18.4118	3.507569038	4.477711538
B,C	10.51034	-17.1241	5.494722691	2.505234248
B,E	14.16	-14.12	0.400935024	7.599978102
C,E	14.54118	-19.9647	0.100246329	7.904722667
A,D	15.43103	-7.67241	6.706977733	3.293016889
A,F	18.1	-3.7	1.521494574	8.478505426
A,L	17.20943	-2.18302	0.178468105	9.821531895
D,G	16	-9	7.236067977	2.763932023
D,I	17.45283	-8.58491	3.351673233	6.648326767
D,G	20.8	-11.4	1.211145618	4.788854382
E,G	21.23077	-14.8462	0.781199215	5.218800785
C,H	23.1843	-21.5666	2.422643158	5.577356842
C,T4	22.81508	-23.6059	7.610640014	0.389359986
A,L	26.10566	-2.56981	0.567515441	5.432484559
D,I	20.8	-11.4	6.974867712	10.26407426
F,L	25.92	-4.44	3.565685425	2.434314575
G,J	27.37931	-11.5517	4.531534395	1.38386101
G,K	27.66972	-14.1009	4.149376445	1.850636402
G,K	34.55046	-16.1651	2.084414215	5.915633144
H,M	34.32432	-19.9459	2.027212152	5.972787848
K,M	34.8	-16.4	2.211145618	5.788854382
K,T3	33.13846	-16.4923	5.736486284	2.263513716
A,L	34.09057	-2.91698	0.915013235	5.084986765
F,L	33.76	-3.32	4.697056275	1.302943725
I,L	33.6	-3.8	4.264911064	1.735088936
J,T1	35.368	-6.976	0.601408556	5.40290806

### 3. Risk calculation table

Properties (1-A, Zone1)		values
shortest distance b/w risk center and path		1.77
Ground Property		0.9
Impact		4620
distance b/w Intersection and starting point		3.765045
distance b/w Intersection and ending point		0.234955
Risk value of path		11520.6
normalized risk value of a path		1.31%
Properties (2-C, Zone2)		values
shortest distance b/w risk center and path		3.58
Ground Property		0.95
Impact		6520
distance b/w Intersection and starting point		7.577709
distance b/w Intersection and ending point		0.422291
Risk value of path		25782
normalized risk value of a path		2.94%
Properties (2-E, Zone2)		values
shortest distance b/w risk center and path		2.51
Ground Property		0.95
Impact		6520
distance b/w Intersection and starting point		1.485728
distance b/w Intersection and ending point		6.514339
Risk value of path		25305.54
normalized risk value of a path		2.88%
Properties (3-E, Zone2)		values
shortest distance b/w risk center and path		1.46
Ground Property		0.95
Impact		6520
distance b/w Intersection and starting point		3.507569
distance b/w Intersection and ending point		4.477712
Risk value of path		26053.44
normalized risk value of a path		2.97%
Properties (B-C, Zone2)		values
shortest distance b/w risk center and path		1.49
Ground Property		0.95
Impact		6520
distance b/w Intersection and starting point		5.494723
distance b/w Intersection and ending point		2.505234
Risk value of path		25429.52
normalized risk value of a path		2.90%

Properties (B-E, Zone2)		values
shortest distance b/w risk center and path		3.60
Ground Property		0.95
Impact		6520
distance b/w Intersection and starting point		0.400935
distance b/w Intersection and ending point		7.599978
Risk value of path		25799.6
normalized risk value of a path		2.94%
Properties (C-E, Zone2)		values
shortest distance b/w risk center and path		3.90
Ground Property		0.95
Impact		6520
distance b/w Intersection and starting point		0.100246
distance b/w Intersection and ending point		7.904723
Risk value of path		26042.96
normalized risk value of a path		2.97%
Properties (A-D, Zone3)		values
shortest distance b/w risk center and path		1.71
Ground Property		1.1
Impact		6340
distance b/w Intersection and starting point		6.706978
distance b/w Intersection and ending point		3.293017
Risk value of path		31789.78
normalized risk value of a path		3.62%
Properties (A-F, Zone3)		values
shortest distance b/w risk center and path		3.48
Ground Property		1.1
Impact		6340
distance b/w Intersection and starting point		1.521495
distance b/w Intersection and ending point		8.478505
Risk value of path		31672.17
normalized risk value of a path		3.61%
Properties (A-L, Zone3)		values
shortest distance b/w risk center and path		4.82
Ground Property		1.1
Impact		6340
distance b/w Intersection and starting point		0.178468
distance b/w Intersection and ending point		9.821532
Risk value of path		32382.08
normalized risk value of a path		3.69%



Properties (D-G, Zone3)	values
shortest distance b/w risk center and path	2.24
Ground Property	1.1
Impact	6340
distance b/w Intersection and starting point	7.236068
distance b/w Intersection and ending point	2.763932
Risk value of path	5939.803
normalized risk value of a path	0.68%
Properties (D-I, Zone3)	values
shortest distance b/w risk center and path	1.65
Ground Property	1.1
Impact	6340
distance b/w Intersection and starting point	3.351673
distance b/w Intersection and ending point	6.648327
Risk value of path	31806.76
normalized risk value of a path	3.63%
Properties (D-G, Zone4)	values
shortest distance b/w risk center and path	1.8
Ground Property	1
Impact	8420
distance b/w Intersection and starting point	1.211146
distance b/w Intersection and ending point	4.788854
Risk value of path	29537.08
normalized risk value of a path	3.37%
Properties (E,G, Zone4)	values
shortest distance b/w risk center and path	2.22
Ground Property	1
Impact	8420
distance b/w Intersection and starting point	0.781199
distance b/w Intersection and ending point	5.218801
Risk value of path	29722.34
normalized risk value of a path	3.39%
Properties (C-H, Zone5)	values
shortest distance b/w risk center and path	1.58
Ground Property	1.05
Impact	10320
distance b/w Intersection and starting point	2.422643
distance b/w Intersection and ending point	5.577357
Risk value of path	44444.3
normalized risk value of a path	5.07%

Properties (C-T4, Zone5)	values
shortest distance b/w risk center and path	3.61
Ground Property	1.05
Impact	10320
distance b/w Intersection and starting point	7.61064
distance b/w Intersection and ending point	0.38936
Risk value of path	45145.44
normalized risk value of a path	5.15%
Properties (A-L, Zone6)	values
shortest distance b/w risk center and path	2.4
Ground Property	1.1
Impact	6250
distance b/w Intersection and starting point	0.567515
distance b/w Intersection and ending point	5.432485
Risk value of path	24422.92
normalized risk value of a path	2.78%
Properties (D-I, Zone6)	values
shortest distance b/w risk center and path	1.1
Ground Property	1.1
Impact	6250
distance b/w Intersection and starting point	6.974868
distance b/w Intersection and ending point	10.26407
Risk value of path	2633.409
normalized risk value of a path	0.30%
Properties (F-L, Zone6)	values
shortest distance b/w risk center and path	0.6
Ground Property	1.1
Impact	6250
distance b/w Intersection and starting point	3.565685
distance b/w Intersection and ending point	2.434315
Risk value of path	24521.64
normalized risk value of a path	2.80%
Properties (G-J, Zone7)	values
shortest distance b/w risk center and path	1.58
Ground Property	0.95
Impact	2630
distance b/w Intersection and starting point	4.531534
distance b/w Intersection and ending point	1.383861
Risk value of path	8695.923
normalized risk value of a path	0.99%

Properties (G-K, Zone7)	values
shortest distance b/w risk center and path	1.15
Ground Property	0.95
Impact	2630
distance b/w Intersection and starting point	4.149376
distance b/w Intersection and ending point	1.850636
Risk value of path	8815.497
normalized risk value of a path	1.00%
Properties (G-K, Zone8)	values
shortest distance b/w risk center and path	1.9
Ground Property	1
Impact	14210
distance b/w Intersection and starting point	2.084414
distance b/w Intersection and ending point	5.915633
Risk value of path	12840.34
normalized risk value of a path	1.46%
Properties (H-M, Zone8)	values
shortest distance b/w risk center and path	1.97
Ground Property	1
Impact	14210
distance b/w Intersection and starting point	2.027212
distance b/w Intersection and ending point	5.972788
Risk value of path	58073.1
normalized risk value of a path	6.62%
Properties (K-M, Zone8)	values
shortest distance b/w risk center and path	1.79
Ground Property	1
Impact	14210
distance b/w Intersection and starting point	2.211146
distance b/w Intersection and ending point	5.788854
Risk value of path	58155.53
normalized risk value of a path	6.63%
Properties (K-T3, Zone8)	values
shortest distance b/w risk center and path	1.74
Ground Property	1
Impact	14210
distance b/w Intersection and starting point	5.736486
distance b/w Intersection and ending point	2.263514
Risk value of path	11790.1
normalized risk value of a path	1.34%

Properties (A-L, Zone9)	values
shortest distance b/w risk center and path	2.1
Ground Property	1.1
Impact	14120
distance b/w Intersection and starting point	0.915013
distance b/w Intersection and ending point	5.084987
Risk value of path	54667.58
<b>normalized risk value of a path</b>	<b>6.23%</b>
Properties (F-L, Zone9)	values
shortest distance b/w risk center and path	1.7
Ground Property	1.1
Impact	14120
distance b/w Intersection and starting point	4.697056
distance b/w Intersection and ending point	1.302944
Risk value of path	54477.29
<b>normalized risk value of a path</b>	<b>6.21%</b>
Properties (I-L, Zone9)	values
shortest distance b/w risk center and path	1.26
Ground Property	1.1
Impact	14120
distance b/w Intersection and starting point	4.264911
distance b/w Intersection and ending point	1.735089
Risk value of path	54692.77
<b>normalized risk value of a path</b>	<b>6.23%</b>
Properties (J-T1, Zone9)	values
shortest distance b/w risk center and path	2.40
Ground Property	1.1
Impact	14120
distance b/w Intersection and starting point	0.601409
distance b/w Intersection and ending point	5.402908
Risk value of path	55150.91
<b>normalized risk value of a path</b>	<b>6.29%</b>
Result	values
multiple risky zones assigned arc	32.15%
A-L: Zone3, Zone6, Zone9	12.71%
D-G: Zone3, Zone4	4.04%
D-I: Zone3, Zone6	3.93%
F-L: Zone6, Zone9	9.00%
G-K: Zone7, Zone8	2.47%
<b>Total risk value for entire map</b>	<b>877310.4</b>

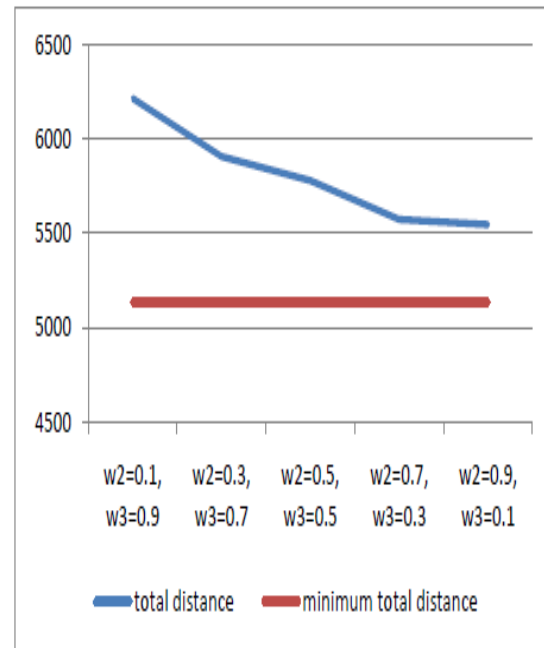
## 4. Multi-objective problems result

### 4.1. Weighted sum approach: Value changing

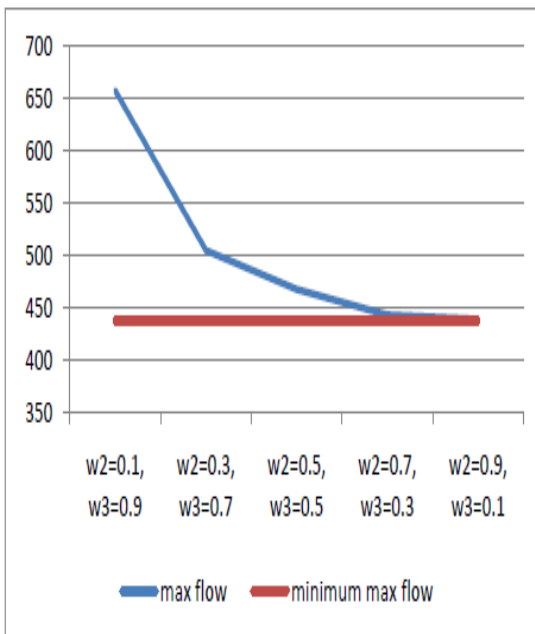
#### 4.1.1. Weighted value for min total distance = 0

weighted value for min dist ( $w_1 = 0$ )			
weighted values	total dist	max flow	total risk
$w_2=0.1, w_3=0.9$	6213.3	657.4	967.74
$w_2=0.3, w_3=0.7$	5905	504.3	1043.28
$w_2=0.5, w_3=0.5$	5777.6	467.3	1103.7
$w_2=0.7, w_3=0.3$	5571.7	442.6	1193.31
$w_2=0.9, w_3=0.1$	5545.1	438.4	1216.12

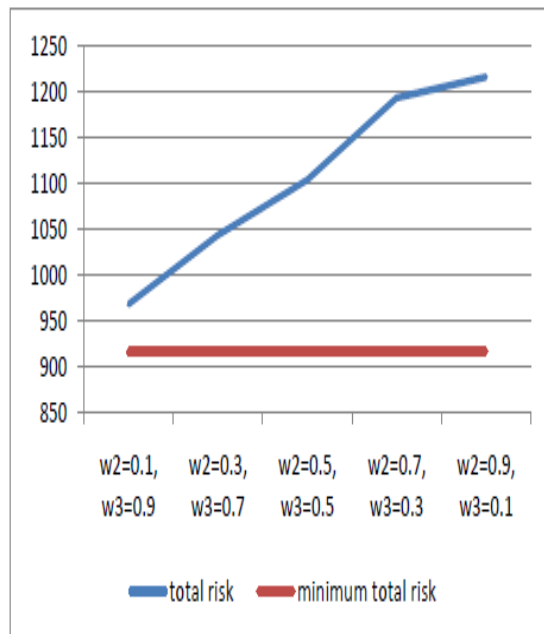
Single objective	Value
min total dist	5135.5
min max flow	437.1
min total risk	915.9



(a) total distance value changes



(b) maximu flow value changes

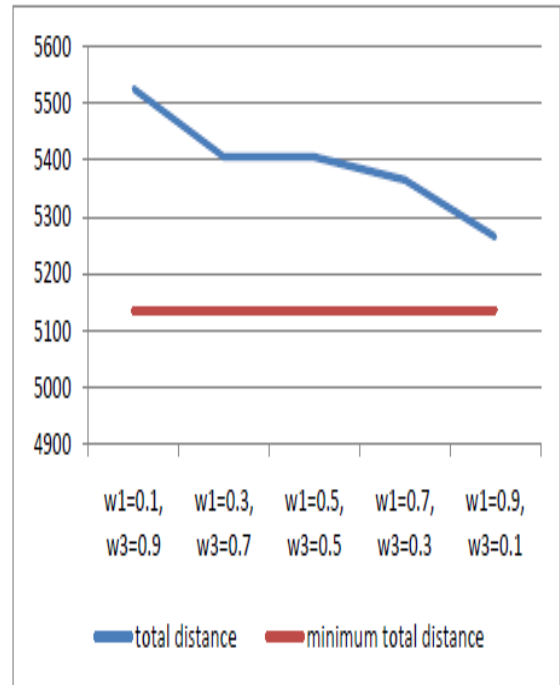


(c) total risk value changes

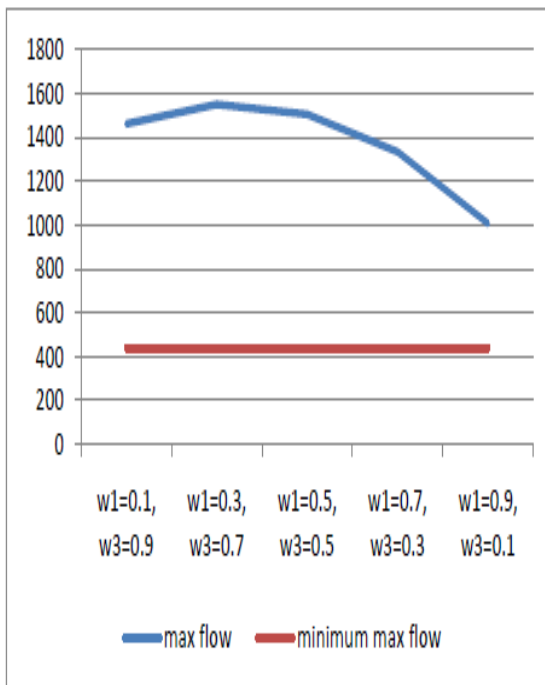
**4.1.2. Weighted value for min max flow =0**

weighted value for min max flow (w2) = 0			
weighted values	total dist	max flow	total risk
w1=0.1, w3=0.9	5525.5	1460	915.9
w1=0.3, w3=0.7	5405.5	1549	920.2
w1=0.5, w3=0.5	5367	1504.5	925.35
w1=0.7, w3=0.3	5265	1333.5	959.55
w1=0.9, w3=0.1	5148	1005.5	1075.35

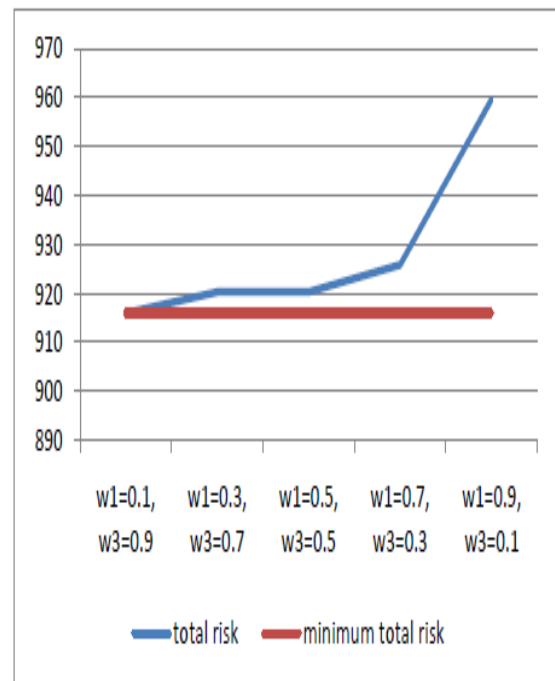
Single objective	Value
min total dist	5135.5
min max flow	437.1
min total risk	915.9



(a) total distance value changes



(b) maximum flow value changes

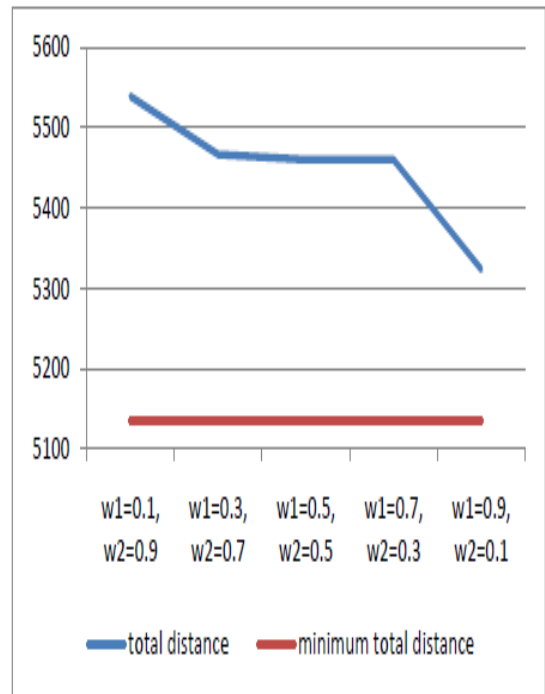


(c) total risk value changes

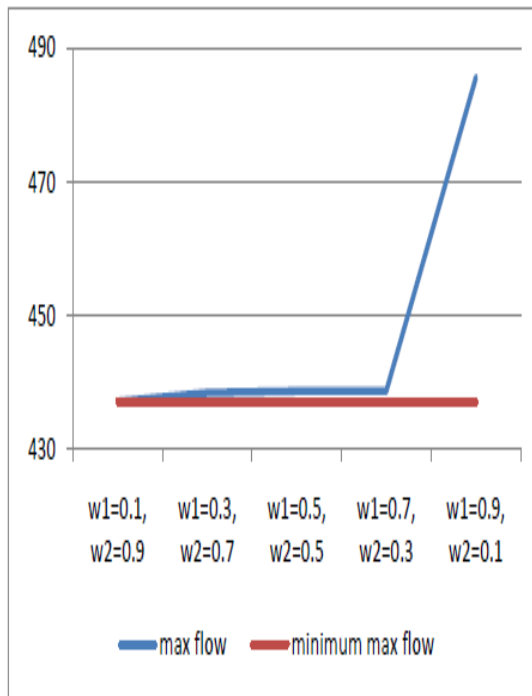
**4.1.3. Weighted value for min total risk = 0**

weighted value for min risk ( $w_3$ ) = 0			
weighted values	total dist	max flow	total risk
w1=0.1, w2=0.9	5539	437.1	1293.18
w1=0.3, w2=0.7	5466.5	438.4	1264.98
w1=0.5, w2=0.5	5459.8	438.7	1264.06
w1=0.7, w2=0.3	5459.8	438.7	1264.06
w1=0.9, w2=0.1	5323.6	485.5	1216.51

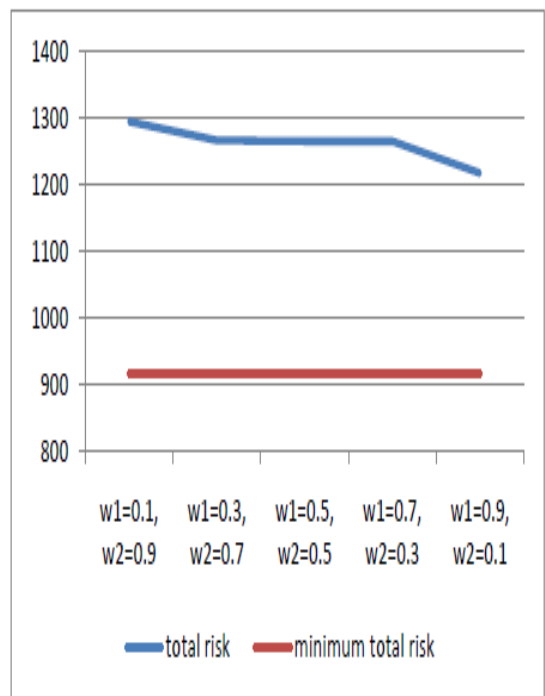
Single objective	Value
min total dist	5135.5
min max flow	437.1
min total risk	915.9



(a) total distance value changes



(b) maximum flow value changes



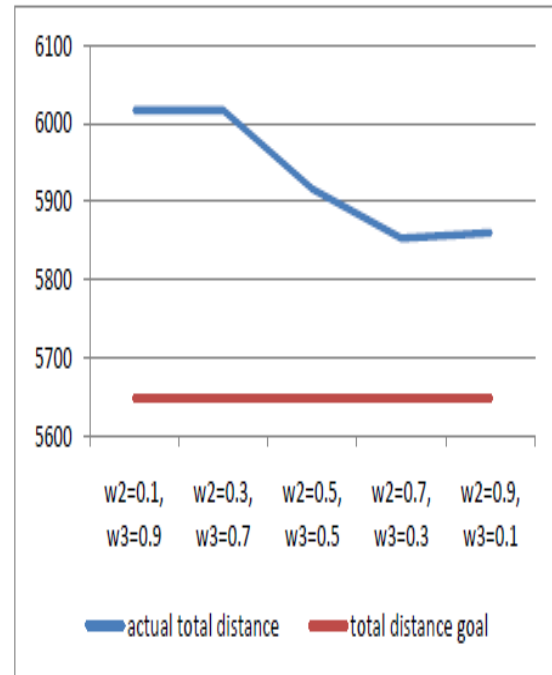
(c) total risk value changes

## 4.2. Goal programming: Value changing

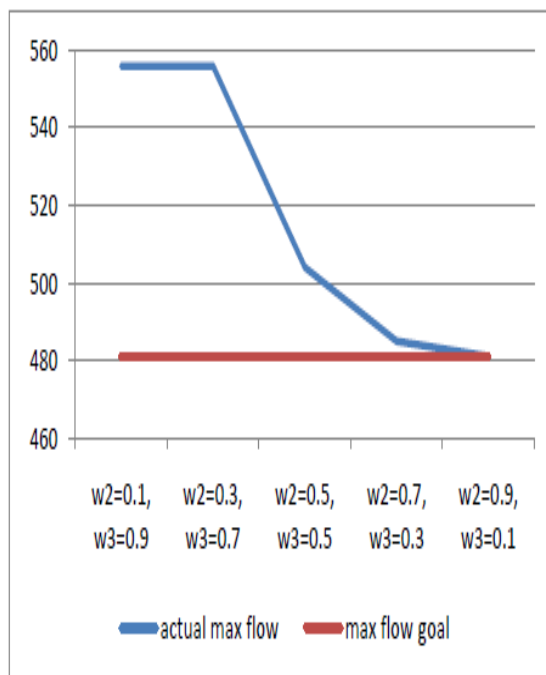
### 4.2.1. Weighted value for min total distance goal = 0

weighted value for min dist goal ( $w_1 = 0$ )			
weighted values	total dist	max flow	total risk
$w_2=0.1, w_3=0.9$	6017	556	1007
$w_2=0.3, w_3=0.7$	6017	556	1007
$w_2=0.5, w_3=0.5$	5916	504	1044
$w_2=0.7, w_3=0.3$	5853	485	1072
$w_2=0.9, w_3=0.1$	5860	481	1082

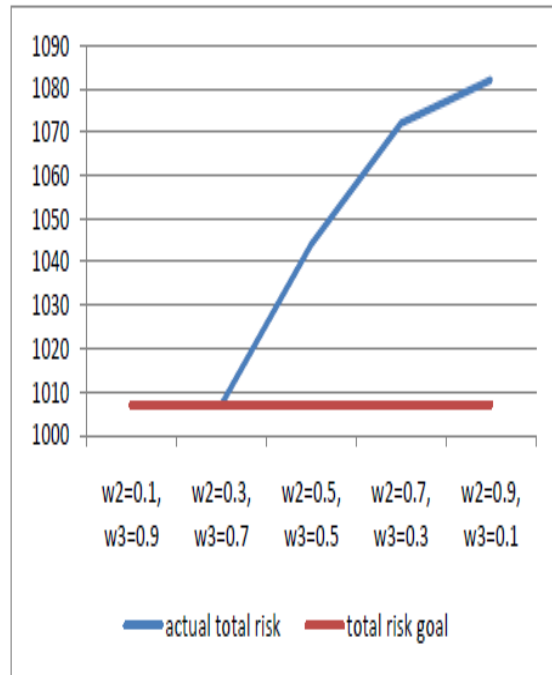
Single objective goal	Value
min total dist	5648
min max flow	481
min total risk	1007



(a) total distance value changes



(b) maximum flow value changes



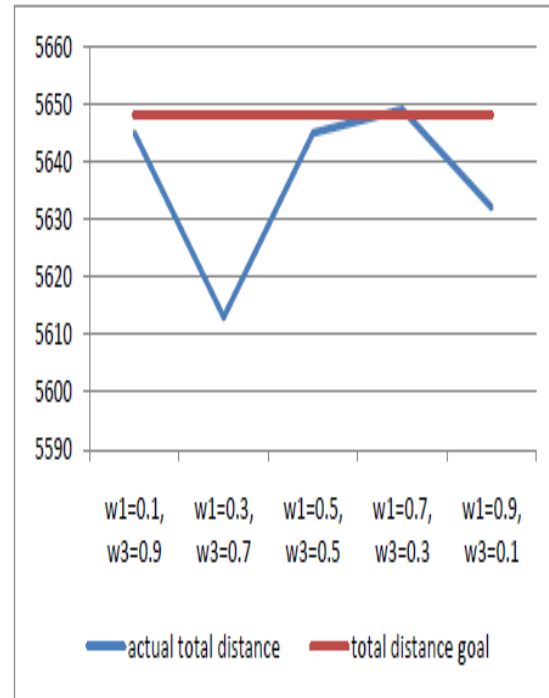
(c) total risk value changes



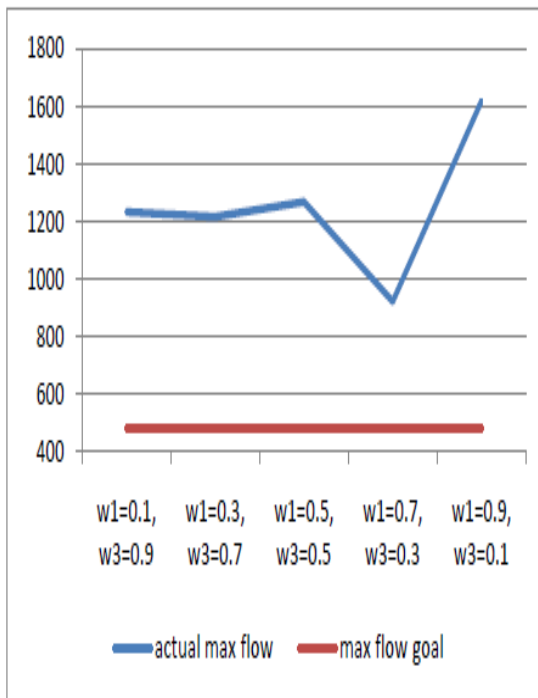
**4.2.2. Weighted value for min max flow goal = 0**

weighted value for min max flow goal( $w_2$ ) = 0			
weighted values	total dist	max flow	total risk
w1=0.1, w3=0.9	5645	1231	979
w1=0.3, w3=0.7	5613	1215	1007
w1=0.5, w3=0.5	5645	1266	994
w1=0.7, w3=0.3	5649	921	1007
w1=0.9, w3=0.1	5632	1615	992

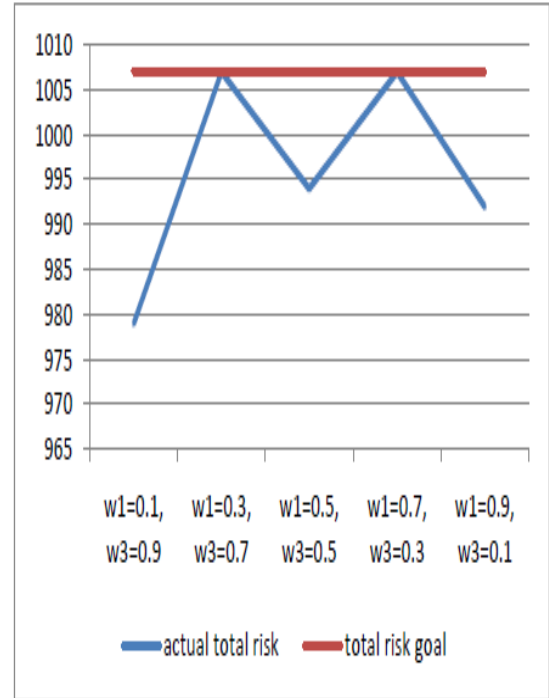
Single objective goal	Value
min total dist	5135.5
min max flow	410.4
min total risk	915.9



(a) total distance value changes



(b) maximum flow value changes

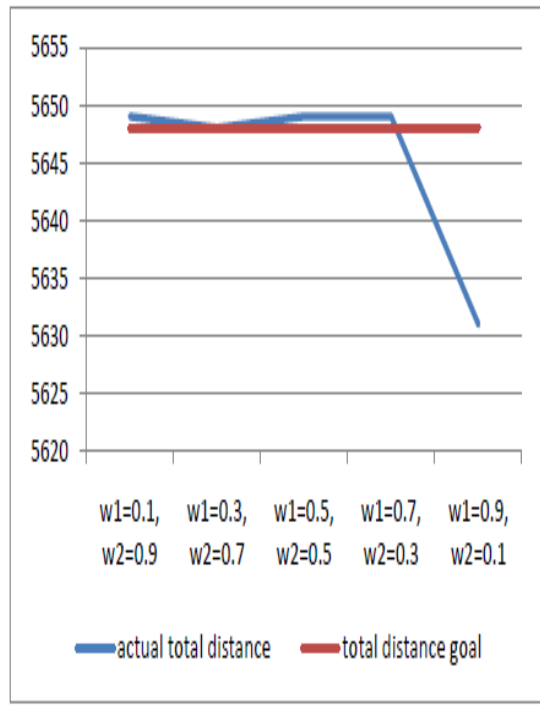


(c) total risk value changes

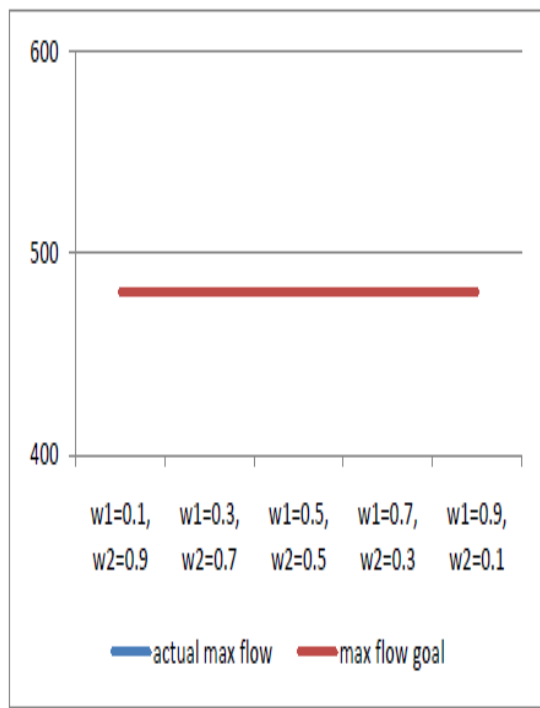
**4.2.3. Weighted value for min total distance goal = 0**

weighted value for min risk goal ( $w_3 = 0$ )			
weighted values	total dist	max flow	total risk
w1=0.1, w2=0.9	5649	481	1101
w1=0.3, w2=0.7	5648	481	1101
w1=0.5, w2=0.5	5649	481	1101
w1=0.7, w2=0.3	5649	481	1101
w1=0.9, w2=0.1	5631	481	1349

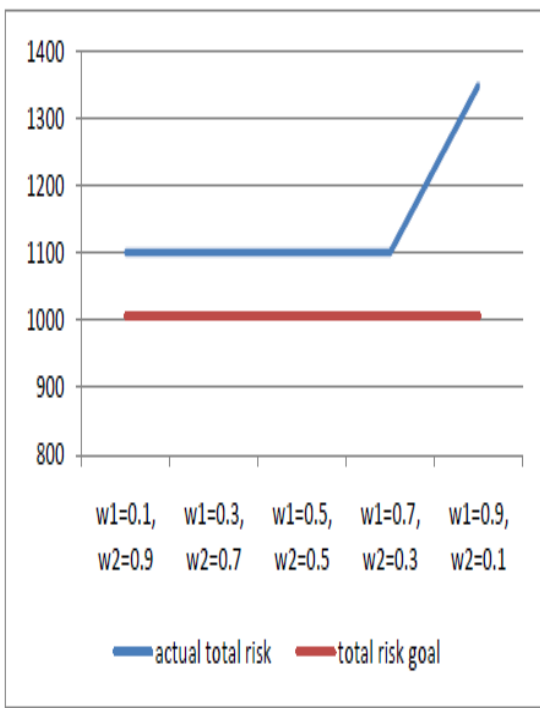
Single objective goal	Value
min total dist	5135.5
min max flow	410.4
min total risk	915.9



(a) total distance value changes



(b) maximum flow value changes



(c) total risk value changes

## 5. MATLAB code

### 5.1. Value iteration

```
% function for showing the analyzing result
%-----
function showDP
% Data for visualizing=
    visiting=0;
    vi=1;
    tstepsize=0;
    tencount=0;
    taccost=0;
    tencost=0;

% Generate information vector
% visitinginfo(vi).vis=[x_coor;y_coor;number of visiting]
for i=1:4
    for j=1:4
        visitinginfo(vi).vis=[i;j;visiting];
        vi=vi+1;
    end
end

SimStruct = SimValueIteration

for i=1:length(SimStruct)
    SimStruct(i).Coordinates
    SimStruct(i).Steps
    SimStruct(i).Encounters
    SimStruct(i).taccost
    SimStruct(i).tencost
    tstepsize=tstepsize+SimStruct(i).Steps;
    tencount=tencount+SimStruct(i).Encounters;
    taccost=taccost+SimStruct(i).taccost;
    tencost=tencost+SimStruct(i).tencost;

    for j=1:size(SimStruct(i).Coordinates,2)
        for k=1:vi-1
            if SimStruct(i).Coordinates(:,j) == [visitinginfo(k).vis(1);visitinginfo(k).vis(2)]
                visitinginfo(k).vis(3)=visitinginfo(k).vis(3)+1;
            end
        end
    end
end

% for visualizing, store the visiting numbers as a matrix form
for g=1:vi-1
    M(visitinginfo(g).vis(1),visitinginfo(g).vis(2))=visitinginfo(g).vis(3);
end
end
M
avgcount=tencount/(length(SimStruct)-1)
avgstepsize=tstepsize/(length(SimStruct)-1)
avgencost=tencost/(length(SimStruct)-1)
avgaccost=taccost/(length(SimStruct)-1)
```

```

% simulation function
%-----
function SimStruct = SimValueIteration

% Parameters
HostileLoc(1).BinMat = [0, 1, 0, 0;
                       0, 1, 1, 0;
                       0, 0, 1, 0;
                       0, 0, 0, 0];

HostileLoc(2).BinMat = [0, 0, 0, 0;
                       0, 1, 0, 0;
                       1, 1, 0, 0;
                       0, 1, 0, 0];

HostileLoc(3).BinMat = [0, 1, 1, 1;
                       0, 0, 0, 1;
                       0, 0, 0, 0;
                       0, 0, 0, 0];
s0=[1 2 3 1; 1 2 1 3];
DestLoc = [4;4];
gamma=1;
StayingCost=2;
MovingCost=1;
EncounterCost=[10;15;20];
EnemyAggression=[1.6;1.8;1.4];
SimLength = 1000;
T=10;

% Call the Value Iteration for individual cases
[V, oa] = ValueIteration(DestLoc, HostileLoc, StayingCost, MovingCost, EncounterCost,
EnemyAggression, T)

% State Generation
h = size(HostileLoc,2);
NumRows = size(HostileLoc(h).BinMat,1);
NumCols = size(HostileLoc(h).BinMat,2);

x=1;
State(x).s = [];
[EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc);
IND = VectorIndexCombination(SizeofPosELoc);

for i=1:NumRows
    for j=1:NumCols
        for k=1:size(IND,1)
            enemycomb=[];
            for e=1:h
                enemycomb=[enemycomb, EnemyPosLoc(e).EPL(:,IND(k,e))];
            end
            State(x).s=[[i;j],enemycomb];
            x=x+1;
        end
    end
end

```

```

end
end

% Simulation
s = s0;
SimCount = 1; %Number of runs
SimStruct(1).Coordinates = [];
SimStruct(1).Steps = 0;
SimStruct(1).Encounters = 0;
SimStruct(1).taccost=0;
SimStruct(1).tencost=0;

while SimCount <= SimLength
    StateIndex(State, s);
    x_star = oa(:,StateIndex(State, s));

    %Update movement coordinates
    SimStruct(SimCount).Coordinates = [SimStruct(SimCount).Coordinates, x_star];

    %Update step counts
    SimStruct(SimCount).Steps = SimStruct(SimCount).Steps + 1;
    SimStruct(SimCount).taccost=SimStruct(SimCount).taccost +
ActionCost(s(:,1),x_star,StayingCost,MovingCost);
    PreviousSimCount = SimCount;
    %Generate next state
    [s, SimCount] = StateTransfer(s0, DestLoc, s, x_star, EnemyAggression, HostileLoc, SimCount);

    %Update encounter counts
    for e=2:h+1
        if s(:,1) == s(:,e)
            SimStruct(SimCount).Encounters = SimStruct(SimCount).Encounters + 1;
        end
    end

    %New run if generated next state is the initial state
    if PreviousSimCount ~=SimCount
        SimStruct(SimCount-1).tencost=SimStruct(SimCount-1).tencost + EnCost(s,EncounterCost); %
update encounter cost for previous iterations
        % initialize SimStruct
        SimStruct(SimCount).Coordinates = [];
        SimStruct(SimCount).Steps = 0;
        SimStruct(SimCount).Encounters = 0;
        SimStruct(SimCount).taccost=0;
        SimStruct(SimCount).tencost=0;
    else
        SimStruct(SimCount).tencost=SimStruct(SimCount).tencost + EnCost(s,EncounterCost); % update
encounter cost
    end
end

% Value Iteration Function
%-----
function [V, oa] = ValueIteration(FinalLoc, HostileLoc, StayingCost, MovingCost, EncounterCost,

```

```

EnemyAggression, T)
% State Generation
h = size(HostileLoc,2);
NumRows = size(HostileLoc(h).BinMat,1);
NumCols = size(HostileLoc(h).BinMat,2);

x=1;
State(x).s = [];
[EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc);
IND = VectorIndexCombination(SizeofPosELoc);

for i=1:NumRows
    for j=1:NumCols
        for k=1:size(IND,1)
            %State(x).s=[i;j];
            enemycomb=[];
            for e=1:h
                enemycomb=[enemycomb, EnemyPosLoc(e).EPL(:,IND(k,e))];
            end
            State(x).s=[[i;j],enemycomb];
            x=x+1;
        end
    end
end
length(State);

% Initializing Values
V = 1000 * ones(length(State),1);

for index =1:length(State)
    if State(index).s(:,1) == FinalLoc
        V(index)=0;
    end
end

% Call the cost function
for l=1:T
    [V,oa]=CostGoUpdate(V,State, IND, HostileLoc, StayingCost, MovingCost,
EncounterCost,EnemyAggression)
end

% Cost function
%-----
function [V,oa]=CostGoUpdate(V_current,State, IND, HostileLoc, StayingCost, MovingCost,
EncounterCost,EnemyAggression);
% This function will give the values and optimal actions based on the state
NumRows = size(HostileLoc(1).BinMat,1);
NumCols = size(HostileLoc(1).BinMat,2);
oa=[];
NumState=size(State,2);
Sindex=size(IND,1);
V=zeros(NumState,1);

for k=1:Sindex

```

```

V(NumState-k)=V_current(NumState-k);
end

%Iteration: Entire states except Destination Point
for i=1:NumState-Sindex
    s=State(i).s;
    CurrentLoc=s(:,1);
    A=ActionspaceI(CurrentLoc,NumRows,NumCols);
    [rows, columns]=size(A);
    TempV=zeros(columns,1);

    % Based on current my location, find out the possible action set A
    % Compare the expect costs among the action set
    for j=1:columns
        PossibleEnemyComb = EnemyMovements(s,HostileLoc,EnemyAggression);
        n=length(PossibleEnemyComb);
        remain=0;
        for pnc=1:n
            NextState=[A(:,j),PossibleEnemyComb(pnc).PEC];
            remain=remain+((PossibleEnemyComb(pnc).PPEC)*(EnCost(NextState,EncounterCost)+(V_current(State
            Index(State,NextState)))));
        end
        TempV(j)=remain+ActionCost(CurrentLoc, A(:,j), StayingCost, MovingCost);
    end

    % Choose the action that has minimum expected cost
    V(i) = min(TempV);
    PossibleAction=[];
    for k=1:columns
        if TempV(k) == V(i)
            PossibleAction=[PossibleAction,A(:,k)];
        end
    end

    % if there are several action that has min cost, choose the action that closest from
    destination point
    if size(PossibleAction,2) > 1
        poa = PossibleAction(:,1);
        m= norm(poa-[NumRows;NumCols]);
        for l=2:size(PossibleAction,2)
            if norm(PossibleAction(:,l)-[NumRows;NumCols]) <= m
                poa = PossibleAction(:,l);
                m=norm(poa-[NumRows;NumCols]);
            end
        end
    else
        poa=PossibleAction;
    end

    oa=[oa,poa];
end

% This function generate my possible action set based on current location

```

```

%-----
function ASI=ActionspaceI(CurrentLoc,NumRows,NumCols)

ASI=[];

for i=-1:1
    for j=-1:1
        a=CurrentLoc+[i;j];
        if a(1)>=1 & a(1)<=NumRows & a(2)>=1 & a(2)<=NumCols
            ASI=[ASI,a];
        end
    end
end

% This function returns the one enemy's possible next position set
%-----
function ASE =ActionspaceE(HostileLoc, CurrentEnemyLoc)

ASE=[];
for i=-1:1
    for j=-1:1
        a=CurrentEnemyLoc+[i;j];
        if a(1) >=1 & a(1) <= size(HostileLoc,1) & a(2) >=1 & a(2) <= size(HostileLoc,2)
            if HostileLoc(a(1),a(2)) == 1
                ASE=[ASE,a];
            end
        end
    end
end

% This function returns a structure for all possible movements of all
% enemies and associated probabilities
%-----
function PossibleEnemyComb=EnemyMovements(s,HostileLoc,EnemyAggression)
h=length(HostileLoc);

EM=[];
EnemyActionCounts = zeros(h,1);

for e=1:h
    s(:,e+1);
    EM(e).Actions = ActionspaceE(HostileLoc(e).BinMat,s(:,e+1));
    EnemyActionCounts(e)=size(EM(e).Actions,2);
    denom=size(EM(e).Actions,2)-1+EnemyAggression(e);
    EM(e).Prob=(1/denom)*ones(size(EM(e).Actions,2),1);
    dist=1000;
    a_star=0;
    for a=1:size(EM(e).Actions,2)
        if norm(EM(e).Actions(a)-s(:,1)) <= dist;
            dist=norm(EM(e).Actions(a)-s(:,1));
            a_star=a;
        end
    end
end
end

```



```

    EM(e).Prob(a_star)=EnemyAggression(e)/denom;
end
for e=1:h
    EM(e).Prob;
end

IND=VectorIndexCombination(EnemyActionCounts);

for i=1:size(IND,1)
    PossibleEnemyComb(i).PEC=[];
    PossibleEnemyComb(i).PPEC=1;
    for j=1:length(HostileLoc)
        PossibleEnemyComb(i).PEC=[PossibleEnemyComb(i).PEC,EM(j).Actions(:,IND(i,j))];
        PossibleEnemyComb(i).PPEC=PossibleEnemyComb(i).PPEC*EM(j).Prob(IND(i,j));
        if i==1
            PossibleEnemyComb(i).CPPEC=PossibleEnemyComb(i).PPEC;
        else
            PossibleEnemyComb(i).CPPEC=PossibleEnemyComb(i-
1).CPPEC+PossibleEnemyComb(i).PPEC;
        end
    end
end

% This function returns the action cost based on my current loc and action
%-----
function c = ActionCost(CurrentLoc, Action, StayingCost, MovingCost)
    CurrentLoc;
    if CurrentLoc == Action
        c=StayingCost;
    else
        c= MovingCost;
    end

% This function returns the encounter cost given state
%-----
function c = EnCost(StatePrime,EncounterCost)

    MyLoc = StatePrime(:,1);
    NumEnemy = size(StatePrime,2)-1;

    c=0;
    i=2;

    while i <=size(StatePrime,2)
        if MyLoc==StatePrime(:,i)
            c=c+EncounterCost(i-1);
        end
        i=i+1;
    end

% Vector index for combinations of enemies

```

```

%-----
function IND = VectorIndexCombination(VectorSizes)
    numVectors = length(VectorSizes);
    if numVectors == 1
        IND = transpose(1:VectorSizes);
    else
        IND_Temp = VectorIndexCombination(VectorSizes(2:numVectors));
        numRows = size(IND_Temp,1);
        IND = [];
        for i=1:VectorSizes(1)
            IND = [IND; i*ones(numRows,1), IND_Temp];
        end
    end
end

% This function returns a index value of a given state
%-----
function index = StateIndex(State,s)
    NumStates = length(State);
    for index=1:NumStates
        if State(index).s==s
            return;
        end
    end
end

% This function generate each enemies' territory infromation and their sizes
%-----
function [EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc)

    h=size(HostileLoc,2);
    NumRows = size(HostileLoc(1).BinMat,1);
    NumCols = size(HostileLoc(1).BinMat,2);

    for e=1:h
        EnemyPosLoc(e).EPL=[];

        for i=1:NumRows
            for j=1:NumCols
                if HostileLoc(e).BinMat(i,j) == 1
                    EnemyPosLoc(e).EPL=[EnemyPosLoc(e).EPL,[i;j]];
                end
            end
        end
        SizeofPosELoc(e)=size(EnemyPosLoc(e).EPL,2);
    end
end

```

## 5.2. 1<sup>st</sup> Approximation

```

% function for showing the analyzing result
%-----

```

```

function showAppDP1
% Data for visualizing
visiting=0;
vi=1;
tstepsize=0;
tencount=0;
taccost=0;
tencost=0;

% Generate information vector
% visitinginfo(vi).vis=[x_coor;y_coor;number of visiting]
for i=1:4
    for j=1:4
        visitinginfo(vi).vis=[i;j;visiting];
        vi=vi+1;
    end
end

SimStruct = AppSimDP1

for i=1:length(SimStruct)
    SimStruct(i).Coordinates
    SimStruct(i).Steps
    SimStruct(i).Encounters
    SimStruct(i).taccost
    SimStruct(i).tencost
    tstepsize=tstepsize+SimStruct(i).Steps;
    tencount=tencount+SimStruct(i).Encounters;
    taccost=taccost+SimStruct(i).taccost;
    tencost=tencost+SimStruct(i).tencost;
    for j=1:size(SimStruct(i).Coordinates,2)
        for k=1:vi-1
            if SimStruct(i).Coordinates(:,j) == [visitinginfo(k).vis(1);visitinginfo(k).vis(2)]
                visitinginfo(k).vis(3)=visitinginfo(k).vis(3)+1;
            end
        end
    end
end

% for visualizing, store the visiting numbers as a matrix form
for g=1:vi-1
    M(visitinginfo(g).vis(1),visitinginfo(g).vis(2))=visitinginfo(g).vis(3);
end

end
M
avgencount=tencount/(length(SimStruct)-1)
avgstepsize=tstepsize/(length(SimStruct)-1)
avgencost=tencost/(length(SimStruct)-1)
avgaccost=taccost/(length(SimStruct)-1)

% simulation function
%-----
function SimStruct = AppSimDP1

```

```

%Parameters
HostileLoc(1).BinMat = [0, 1, 0, 0;
                       0, 1, 1, 0;
                       0, 0, 1, 0;
                       0, 0, 0, 0];

HostileLoc(2).BinMat = [0, 0, 0, 0;
                       0, 1, 0, 0;
                       1, 1, 0, 0;
                       0, 1, 0, 0];

HostileLoc(3).BinMat = [0, 1, 1, 1;
                       0, 0, 0, 1;
                       0, 0, 0, 0;
                       0, 0, 0, 0];
s0=[1 2 3 1; 1 2 1 3];
DestLoc = [4;4];

gamma=1;
StayingCost=2;
MovingCost=1;
EncounterCost=[10;15;20];
EnemyAggression=[1.6;1.8;1.4];

SimLength = 1000;
T=10;

% Call the Value Iteration for individual cases
for i=1:length(HostileLoc)
    [SubState(i).ss, V(i).v, oa(i).oa] = ValueIteration(DestLoc, HostileLoc(i).BinMat, StayingCost,
    MovingCost, EncounterCost(i), EnemyAggression(i), T);
end

% Generate States: Actual state generation
h = size(HostileLoc,2);
NumRows = size(HostileLoc(h).BinMat,1);
NumCols = size(HostileLoc(h).BinMat,2);

x=1;
State(x).s = [];
[EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc);
IND = VectorIndexCombination(SizeofPosELoc);

for i=1:NumRows
    for j=1:NumCols
        for k=1:size(IND,1)
            enemycomb=[];
            for e=1:h
                enemycomb=[enemycomb, EnemyPosLoc(e).EPL(:,IND(k,e))];
            end
            State(x).s=[[i;j],enemycomb];
            x=x+1;
        end
    end
end
end

```

```

% Simulation
s = s0;
SimCount = 1; %Number of runs

SimStruct(1).Coordinates = [];
SimStruct(1).Steps = 0;
SimStruct(1).Encounters = 0;
SimStruct(1).taccost=0;
SimStruct(1).tencost=0;

while SimCount <= SimLength
    StateIndex(State, s);
    subx_star=[];
    subV_star=[];
    for i=1:h
        substate(i).ss=[s(:,1),s(:,i+1)];% individual states
        subx_star = [subx_star,oa(i).oa(:,StateIndex(SubState(i).ss,substate(i).ss))]; %
        optimal actions at the individual states
        subV_star = [subV_star, V(i).v(StateIndex(SubState(i).ss,substate(i).ss))]; % V
        value at the individual states
    end

    [minv,v_stari]=min(subV_star); % find minimum V value
    x_star=subx_star(:,v_stari); % optimal action at the given state is the optimal action at the
    individual state which has minimum V value

    %Update movement coordinates
    SimStruct(SimCount).Coordinates = [SimStruct(SimCount).Coordinates, x_star];

    %Update step counts, actioncost
    SimStruct(SimCount).Steps = SimStruct(SimCount).Steps +1;
    SimStruct(SimCount).taccost=SimStruct(SimCount).taccost +
    ActionCost(s(:,1),x_star,StayingCost,MovingCost);
    PreviousSimCount = SimCount;
    %Generate next state
    [s, SimCount] = StateTransfer(s0, DestLoc, s, x_star, EnemyAggression, HostileLoc,
    SimCount);

    %Update encounter counts
    for e=2:h+1
        if s(:,1) == s(:,e)
            SimStruct(SimCount).Encounters = SimStruct(SimCount).Encounters +1;
        end
    end

    %New run if generated next state is the initial state
    if PreviousSimCount ~=SimCount
        SimStruct(SimCount-1).tencost=SimStruct(SimCount-1).tencost +
        EnCost(s,EncounterCost); % update encounter cost for previous iterations
        % initialize SimStruct
        SimStruct(SimCount).Coordinates = [];
        SimStruct(SimCount).Steps = 0;
        SimStruct(SimCount).Encounters = 0;
    end
end

```

```

        SimStruct(SimCount).taccost=0;
        SimStruct(SimCount).tencost=0;
    else
        SimStruct(SimCount).tencost=SimStruct(SimCount).tencost + EnCost(s,EncounterCost);
        % update encounter cost
    end
end

end

% Value Iteration Function
%-----
function [State, V, oa] = ValueIteration(FinalLoc, HostileLoc, StayingCost, MovingCost,
EncounterCost, EnemyAggression, T)
% State Generation
h = 1;
NumRows = size(HostileLoc,1);
NumCols = size(HostileLoc,2);

x=1;
State(x).s = [];
[EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc);
IND = VectorIndexCombination(SizeofPosELoc);

for i=1:NumRows
    for j=1:NumCols
        for k=1:size(IND,1)
            enemycomb=[];
            for e=1:h
                enemycomb=[enemycomb, EnemyPosLoc(e).EPL(:,IND(k,e))];
            end
            State(x).s=[[i;j],enemycomb];
            x=x+1;
        end
    end
end
length(State);

% Initializing Values
V = 1000 * ones(length(State),1);

for index =1:length(State)
    if State(index).s(:,1) == FinalLoc
        V(index)=0;
    end
end

% Call the cost function
for l=1:T
    [V,oa]=CostGoUpdate(V,State, IND, HostileLoc, StayingCost, MovingCost,
EncounterCost,EnemyAggression)
end

% Cost function

```

```

%-----
function [V,oa]=CostGoUpdate(V_current,State, IND, HostileLoc, StayingCost, MovingCost,
EncounterCost,EnemyAggression);
% This function will give the values and optimal actions based on the state
NumRows = size(HostileLoc,1);
NumCols = size(HostileLoc,2);
oa=[];
NumState=size(State,2);
Sindex=size(IND,1);
V=zeros(NumState,1);

for k=1:Sindex
    V(NumState-k)=V_current(NumState-k);
end

%Iteration: Entire states except Destination Point
for i=1:NumState-Sindex
    s=State(i).s;
    CurrentLoc=s(:,1);
    A=ActionspaceI(CurrentLoc,NumRows,NumCols);
    [rows, columns]=size(A);
    TempV=zeros(columns,1);

    % Based on current my location, find out the possible action set A
    % Compare the expect costs among the action set
    for j=1:columns
        PossibleEnemyComb = EnemyMovements(s,HostileLoc,EnemyAggression);
        n=length(PossibleEnemyComb);
        remain=0;
        for pnc=1:n
            NextState=[A(:,j),PossibleEnemyComb(pnc).PEC];

remain=remain+((PossibleEnemyComb(pnc).PPEC)*(EnCost(NextState,EncounterCost)+(V_cur
rent(StateIndex(State,NextState)))));
            end
            TempV(j)=remain+ActionCost(CurrentLoc, A(:,j), StayingCost, MovingCost);
        end

        % Choose the action that has minimum expected cost
        V(i) = min(TempV);
        PossibleAction=[];
        for k=1:columns
            if TempV(k) == V(i)
                PossibleAction=[PossibleAction,A(:,k)];
            end
        end

        % if there are several action that has min cost, choose the action that closest
        from destination point
        if size(PossibleAction,2) > 1
            poa = PossibleAction(:,1);
            m = norm(poa-[NumRows;NumCols]);
            for l=2:size(PossibleAction,2)
                if norm(PossibleAction(:,l)-[NumRows;NumCols]) <= m
                    poa = PossibleAction(:,l);
                end
            end
        end
    end
end

```

```

        m = norm(poa-[NumRows;NumCols]);
    end
end
else
    poa = PossibleAction;
end
oa=[oa,poa];
end

% This function generate my possible action set based on current location
%-----
function ASI=ActionspaceI(CurrentLoc,NumRows,NumCols)

    ASI=[];

    for i=-1:1
        for j=-1:1
            a=CurrentLoc+[i;j];
            if a(1)>=1 & a(1)<=NumRows & a(2)>=1 & a(2)<=NumCols
                ASI=[ASI,a];
            end
        end
    end

% This function returns the one enemy's possible next position set
%-----
function ASE =ActionspaceE(HostileLoc, CurrentEnemyLoc)

    ASE=[];
    for i=-1:1
        for j=-1:1
            a=CurrentEnemyLoc+[i;j];
            if a(1) >=1 & a(1) <= size(HostileLoc,1) & a(2) >=1 & a(2) <= size(HostileLoc,2)
                if HostileLoc(a(1),a(2)) == 1
                    ASE=[ASE,a];
                end
            end
        end
    end

% This function returns a structure for all possible movements of all enemies and associated
probabilities
%-----
function PossibleEnemyComb=EnemyMovements(s,HostileLoc,EnemyAggression)
    h=1;

    EM=[];
    EnemyActionCounts = zeros(h,1);

    for e=1:h
        s(:,e+1);
        EM(e).Actions = ActionspaceE(HostileLoc,s(:,e+1));
    end
end

```



```

EnemyActionCounts(e)=size(EM(e).Actions,2);
denom=size(EM(e).Actions,2)-1+EnemyAggression(e);
EM(e).Prob=(1/denom)*ones(size(EM(e).Actions,2),1);
dist=1000;
a_star=0;
for a=1:size(EM(e).Actions,2)
    if norm(EM(e).Actions(a)-s(:,1)) <= dist;
        dist=norm(EM(e).Actions(a)-s(:,1));
        a_star=a;
    end
end
EM(e).Prob(a_star)=EnemyAggression(e)/denom;
end
for e=1:h
    EM(e).Prob;
end

IND=VectorIndexCombination(EnemyActionCounts);

for i=1:size(IND,1)
    PossibleEnemyComb(i).PEC=[];
    PossibleEnemyComb(i).PPEC=1;
    for j=1:h
        PossibleEnemyComb(i).PEC=[PossibleEnemyComb(i).PEC,EM(j).Actions(:,IND(i,j))];
        PossibleEnemyComb(i).PPEC=PossibleEnemyComb(i).PPEC*EM(j).Prob(IND(i,j));
        if i==1
            PossibleEnemyComb(i).CPPEC=PossibleEnemyComb(i).PPEC;
        else
            PossibleEnemyComb(i).CPPEC=PossibleEnemyComb(i-1).CPPEC+PossibleEnemyComb(i).PPEC;
        end
    end
end

% This function returns the action cost based on my current loc and action
%-----
function c = ActionCost(CurrentLoc, Action, StayingCost, MovingCost)
    CurrentLoc;
    if CurrentLoc == Action
        c=StayingCost;
    else
        c= MovingCost;
    end

% This function returns the encounter cost given state
%-----
function c = EnCost(StatePrime,EncounterCost)

    MyLoc = StatePrime(:,1);
    NumEnemy = size(StatePrime,2)-1;

    c=0;

```

```

i=2;

while i <=size(StatePrime,2)
    if MyLoc==StatePrime(:,i)
        c=c+EncounterCost(i-1);
    end
    i=i+1;
end

% Vector index for combinations of enemies
%-----
function IND = VectorIndexCombination(VectorSizes)
    numVectors = length(VectorSizes);
    if numVectors == 1
        IND = transpose(1:VectorSizes);
    else
        IND_Temp = VectorIndexCombination(VectorSizes(2:numVectors));
        numRows = size(IND_Temp,1);
        IND = [];
        for i=1:VectorSizes(1)
            IND = [IND; i*ones(numRows,1), IND_Temp];
        end
    end
end

% This function returns a index value of a given state
%-----
function index = StateIndex(State,s)
    NumStates = length(State);
    for index=1:NumStates
        if State(index).s==s
            return;
        end
    end
end

% This function generate each enemies' territory infromation and their sizes
%-----
function [EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc)

    h=1;
    NumRows = size(HostileLoc,1);
    NumCols = size(HostileLoc,2);

    for e=1:h
        EnemyPosLoc(e).EPL=[];

        for i=1:NumRows
            for j=1:NumCols
                if HostileLoc(i,j) == 1
                    EnemyPosLoc(e).EPL=[EnemyPosLoc(e).EPL,[i,j]];
                end
            end
        end
    end
end

```

```
SizeofPosELoc(e)=size(EnemyPosLoc(e).EPL,2);  
end
```

### 5.3. 2<sup>nd</sup> Approximation

```
% function for showing the analuzing result  
%-----  
function showAppDP2  
% Data for visualizing  
visiting=0;
```

```

vi=1;
tstepsize=0;
tencount=0;
taccost=0;
tencost=0;

% Generate information vector
% visitinginfo(vi).vis=[x_coor;y_coor;number of visiting]
for i=1:4
    for j=1:4
        visitinginfo(vi).vis=[i;j;visiting];
        vi=vi+1;
    end
end

SimStruct = AppSimDP2

for i=1:length(SimStruct)

    SimStruct(i).Coordinates
    SimStruct(i).Steps
    SimStruct(i).Encounters
    SimStruct(i).taccost
    SimStruct(i).tencost
    tstepsize=tstepsize+SimStruct(i).Steps;
    tencount=tencount+SimStruct(i).Encounters;
    taccost=taccost+SimStruct(i).taccost;
    tencost=tencost+SimStruct(i).tencost;

    for j=1:size(SimStruct(i).Coordinates,2)
        for k=1:vi-1
            if SimStruct(i).Coordinates(:,j) == [visitinginfo(k).vis(1);visitinginfo(k).vis(2)]
                visitinginfo(k).vis(3)=visitinginfo(k).vis(3)+1;
            end
        end
    end

% for visualizing, store the visiting numbers as a matrix form
for g=1:vi-1
    M(visitinginfo(g).vis(1),visitinginfo(g).vis(2))=visitinginfo(g).vis(3);
end

end
M
avgencount=tencount/(length(SimStruct)-1)
avgstepsize=tstepsize/(length(SimStruct)-1)
avgencost=tencost/(length(SimStruct)-1)
avgaccost=taccost/(length(SimStruct)-1)

%-----
function SimStruct = AppSimDP2

%Parameters

```

```

HostileLoc(1).BinMat = [0, 1, 0, 0;
                      0, 1, 1, 0;
                      0, 0, 1, 0;
                      0, 0, 0, 0];

HostileLoc(2).BinMat = [0, 0, 0, 0;
                      0, 1, 0, 0;
                      1, 1, 0, 0;
                      0, 1, 0, 0];

HostileLoc(3).BinMat = [0, 1, 1, 1;
                      0, 0, 0, 1;
                      0, 0, 0, 0;
                      0, 0, 0, 0];
s0=[1 2 3 1 ;1 2 1 3];
DestLoc = [4;4];
gamma=1;
StayingCost=2;
MovingCost=1;
EncounterCost=[10;15;20];
EnemyAggression=[1.6;1.8;1.4];
SimLength = 1000;
T=10;

% Call the Value Iteration for individual cases
for i=1:length(HostileLoc)
    [SubState(i).ss, V(i).v, oa(i).oa] = ValueIteration(DestLoc, HostileLoc(i).BinMat, StayingCost,
MovingCost, EncounterCost(i), EnemyAggression(i), T)
end

% Generate States: Actual state generation
h = size(HostileLoc,2);
NumRows = size(HostileLoc(h).BinMat,1);
NumCols = size(HostileLoc(h).BinMat,2);

x=1;
State(x).s = [];
[EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc);
IND = VectorIndexCombination(SizeofPosELoc);

for i=1:NumRows
    for j=1:NumCols
        for k=1:size(IND,1)
            enemycomb=[];
            for e=1:h
                enemycomb=[enemycomb, EnemyPosLoc(e).EPL(:,IND(k,e))];
            end
            State(x).s=[[i;j],enemycomb];
            x=x+1;
        end
    end
end

% Simulation

```

```

s = s0;
SimCount = 1; %Number of runs

SimStruct(1).Coordinates = [];
SimStruct(1).Steps = 0;
SimStruct(1).Encounters = 0;
SimStruct(1).taccost=0;
SimStruct(1).tencost=0;

while SimCount <= SimLength
    StateIndex(State, s);
    subx_star=[];
    subV_star=[];
    for i=1:h
        substate(i).ss=[s(:,1),s(:,i+1)];% individual state
        subx_star = [subx_star,oa(i).oa(:,StateIndex(SubState(i).ss,substate(i).ss))]; %
    optimal actions at the individual states
        subV_star = [subV_star, V(i).v(StateIndex(SubState(i).ss,substate(i).ss))]; % V
    values at the individual states
    end

        % calculate cost function
        % my possible action set = optimal actions at the individual states
        % V value for actual state = summation of v values at the generated next individual states
    for j=1:size(subx_star,2)
        PossibleEnemyComb = EnemyMovements(s,HostileLoc,EnemyAggression);
        n=length(PossibleEnemyComb);
        remain=0;
        for pnc=1:n
            NextState=[subx_star(:,j),PossibleEnemyComb(pnc).PEC];
            V_current=0;
            for k=1:size(NextState,2)-1
                subnextstate(k).sns=[NextState(:,1),NextState(:,k+1)];
                vsub(k).vs= V(k).v(StateIndex(SubState(i).ss,subnextstate(k).sns));
                V_current=V_current+vsub(k).vs;
            end

remain=remain+((PossibleEnemyComb(pnc).PPEC)*(EnCost(NextState,EncounterCost)+(V_cur
rent)));
        end
        CurrentLoc=s(:,1);
        tempV(j)=remain+ActionCost(CurrentLoc, subx_star(:,j), StayingCost, MovingCost);
    end

        % my action at the given state = action that has min cost value among the possible action
    set
    [minv,v_stari]=min(tempV);
    x_star=subx_star(:,v_stari);

    %Update movement coordinates
    SimStruct(SimCount).Coordinates = [SimStruct(SimCount).Coordinates, x_star];

    %Update step counts
    SimStruct(SimCount).Steps = SimStruct(SimCount).Steps +1;
    SimStruct(SimCount).taccost=SimStruct(SimCount).taccost +

```

```

ActionCost(s(:,1),x_star,StayingCost,MovingCost);

    PreviousSimCount = SimCount;
    [s, SimCount] = StateTransfer(s0, DestLoc, s, x_star, EnemyAggression, HostileLoc,
SimCount);

    %Update encounter counts
    for e=2:h+1
        if s(:,1) == s(:,e)
            SimStruct(SimCount).Encounters = SimStruct(SimCount).Encounters +1;
        end
    end

    %New run if generated next state is the initial state
    if PreviousSimCount ~=SimCount
        SimStruct(SimCount-1).tencost=SimStruct(SimCount-1).tencost +
EnCost(s,EncounterCost); % update encounter cost for previous iterations
        % initialize SimStruct
        SimStruct(SimCount).Coordinates = [];
        SimStruct(SimCount).Steps = 0;
        SimStruct(SimCount).Encounters = 0;
        SimStruct(SimCount).taccost=0;
        SimStruct(SimCount).tencost=0;
    else
        SimStruct(SimCount).tencost=SimStruct(SimCount).tencost + EnCost(s,EncounterCost);
        % update encounter cost
    end
end

% Value Iteration Function
%-----
function [State, V, oa] = ValueIteration(FinalLoc, HostileLoc, StayingCost, MovingCost,
EncounterCost, EnemyAggression, T)
% State Generation
h = 1;
NumRows = size(HostileLoc,1);
NumCols = size(HostileLoc,2);

x=1;
State(x).s = [];
[EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc);
IND = VectorIndexCombination(SizeofPosELoc);

for i=1:NumRows
    for j=1:NumCols
        for k=1:size(IND,1)
            enemycomb=[];
            for e=1:h
                enemycomb=[enemycomb, EnemyPosLoc(e).EPL(:,IND(k,e))];
            end
            State(x).s=[[i;j],enemycomb];
            x=x+1;
        end
    end
end
end

```

```

length(State);

% Initializing Values
V = 1000 * ones(length(State),1);

for index =1:length(State)
    if State(index).s(:,1) == FinalLoc
        V(index)=0;
    end
end

% Call the cost function
for l=1:T
    [V,oa]=CostGoUpdate(V,State, IND, HostileLoc, StayingCost, MovingCost,
EncounterCost,EnemyAggression)
end

% Cost function
%-----
function [V,oa]=CostGoUpdate(V_current,State, IND, HostileLoc, StayingCost, MovingCost,
EncounterCost,EnemyAggression);
% This function will give the values and optimal actions based on the state
NumRows = size(HostileLoc,1);
NumCols = size(HostileLoc,2);
oa=[];
NumState=size(State,2);
Sindex=size(IND,1);
V=zeros(NumState,1);

for k=1:Sindex
    V(NumState-k)=V_current(NumState-k);
end

% Iteration: Entire states except Destination Point
for i=1:NumState-Sindex
    s=State(i).s;
    CurrentLoc=s(:,1);
    A=ActionspaceI(CurrentLoc,NumRows,NumCols);
    [rows, columns]=size(A);
    TempV=zeros(columns,1);

    % Based on current my location, find out the possible action set A
    % Compare the expect costs among the action set
    for j=1:columns
        PossibleEnemyComb = EnemyMovements(s,HostileLoc,EnemyAggression);
        n=length(PossibleEnemyComb);
        remain=0;
        for pnc=1:n
            NextState=[A(:,j),PossibleEnemyComb(pnc).PEC];

            remain=remain+((PossibleEnemyComb(pnc).PPEC)*(EnCost(NextState,EncounterCost)+(V_cur
rent(StateIndex(State,NextState))))));
        end
        TempV(j)=remain+ActionCost(CurrentLoc, A(:,j), StayingCost, MovingCost);
    end
end

```



```

end

% Choose the action that has minimum expected cost
V(i) = min(TempV);
PossibleAction=[];
for k=1:columns
    if TempV(k) == V(i)
        PossibleAction=[PossibleAction,A(:,k)];
    end
end

% if there are several action that has min cost, choose the action that closest
from destination point
if size(PossibleAction,2) > 1
    poa = PossibleAction(:,1);
    m = norm(poa-[NumRows;NumCols]);
    for l=2:size(PossibleAction,2)
        if norm(PossibleAction(:,l)-[NumRows;NumCols]) <= m
            poa = PossibleAction(:,l);
            m = norm(poa-[NumRows;NumCols]);
        end
    end
else
    poa = PossibleAction;
end
oa=[oa,poa];
end

% This function generate my possible action set based on current location
%-----
function ASI=ActionspaceI(CurrentLoc,NumRows,NumCols)

ASI=[];

for i=-1:1
    for j=-1:1
        a=CurrentLoc+[i;j];
        if a(1)>=1 & a(1)<=NumRows & a(2)>=1 & a(2)<=NumCols
            ASI=[ASI,a];
        end
    end
end

% This function returns the one enemy's possible next position set
%-----
function ASE =ActionspaceE(HostileLoc, CurrentEnemyLoc)

ASE=[];
for i=-1:1
    for j=-1:1
        a=CurrentEnemyLoc+[i;j];
        if a(1) >=1 & a(1) <= size(HostileLoc,1) & a(2) >=1 & a(2) <= size(HostileLoc,2)
            if HostileLoc(a(1),a(2)) == 1

```

```

        ASE=[ASE,a];
    end
end
end
end

% This function returns a structure for all possible movements of all enemies and associated
probabilities
%-----
function PossibleEnemyComb=EnemyMovements(s,HostileLoc,EnemyAggression)
    h=1;

    EM=[];
    EnemyActionCounts = zeros(h,1);

    for e=1:h
        s(:,e+1);
        EM(e).Actions = ActionspaceE(HostileLoc,s(:,e+1));
        EnemyActionCounts(e)=size(EM(e).Actions,2);
        denom=size(EM(e).Actions,2)-1+EnemyAggression(e);
        EM(e).Prob=(1/denom)*ones(size(EM(e).Actions,2),1);
        dist=1000;
        a_star=0;
        for a=1:size(EM(e).Actions,2)
            if norm(EM(e).Actions(a)-s(:,1)) <= dist;
                dist=norm(EM(e).Actions(a)-s(:,1));
                a_star=a;
            end
        end
        EM(e).Prob(a_star)=EnemyAggression(e)/denom;
    end
    for e=1:h
        EM(e).Prob;
    end

    IND=VectorIndexCombination(EnemyActionCounts);

    for i=1:size(IND,1)
        PossibleEnemyComb(i).PEC=[];
        PossibleEnemyComb(i).PPEC=1;
        for j=1:h
            PossibleEnemyComb(i).PEC=[PossibleEnemyComb(i).PEC,EM(j).Actions(:,IND(i,j))];
            PossibleEnemyComb(i).PPEC=PossibleEnemyComb(i).PPEC*EM(j).Prob(IND(i,j));
            if i==1
                PossibleEnemyComb(i).CPPEC=PossibleEnemyComb(i).PPEC;
            else
                PossibleEnemyComb(i).CPPEC=PossibleEnemyComb(i-
1).CPPEC+PossibleEnemyComb(i).PPEC;
            end
        end
    end
end

```

```

% This function returns the action cost based on my current loc and action
%-----
function c = ActionCost(CurrentLoc, Action, StayingCost, MovingCost)
    CurrentLoc;
    if CurrentLoc == Action
        c=StayingCost;
    else
        c= MovingCost;
    end

% This function returns the encounter cost given state
%-----
function c = EnCost(StatePrime,EncounterCost)

    MyLoc = StatePrime(:,1);
    NumEnemy = size(StatePrime,2)-1;

    c=0;
    i=2;

    while i <=size(StatePrime,2)
        if MyLoc==StatePrime(:,i)
            c=c+EncounterCost(i-1);
        end
        i=i+1;
    end

% Vector index for combinations of enemies
%-----
function IND = VectorIndexCombination(VectorSizes)
    numVectors = length(VectorSizes);
    if numVectors == 1
        IND = transpose(1:VectorSizes);
    else
        IND_Temp = VectorIndexCombination(VectorSizes(2:numVectors));
        numRows = size(IND_Temp,1);
        IND = [];
        for i=1:VectorSizes(1)
            IND = [IND; i*ones(numRows,1), IND_Temp];
        end
    end

% This function returns a index value of a given state
%-----
function index = StateIndex(State,s)
    NumStates = length(State);
    for index=1:NumStates
        if State(index).s==s
            return;
        end
    end
end

```

```

% This function generate each enemies' territory information and their sizes
%-----
function [EnemyPosLoc, SizeofPosELoc] = EnemyComb(HostileLoc)

h=1;
NumRows = size(HostileLoc,1);
NumCols = size(HostileLoc,2);

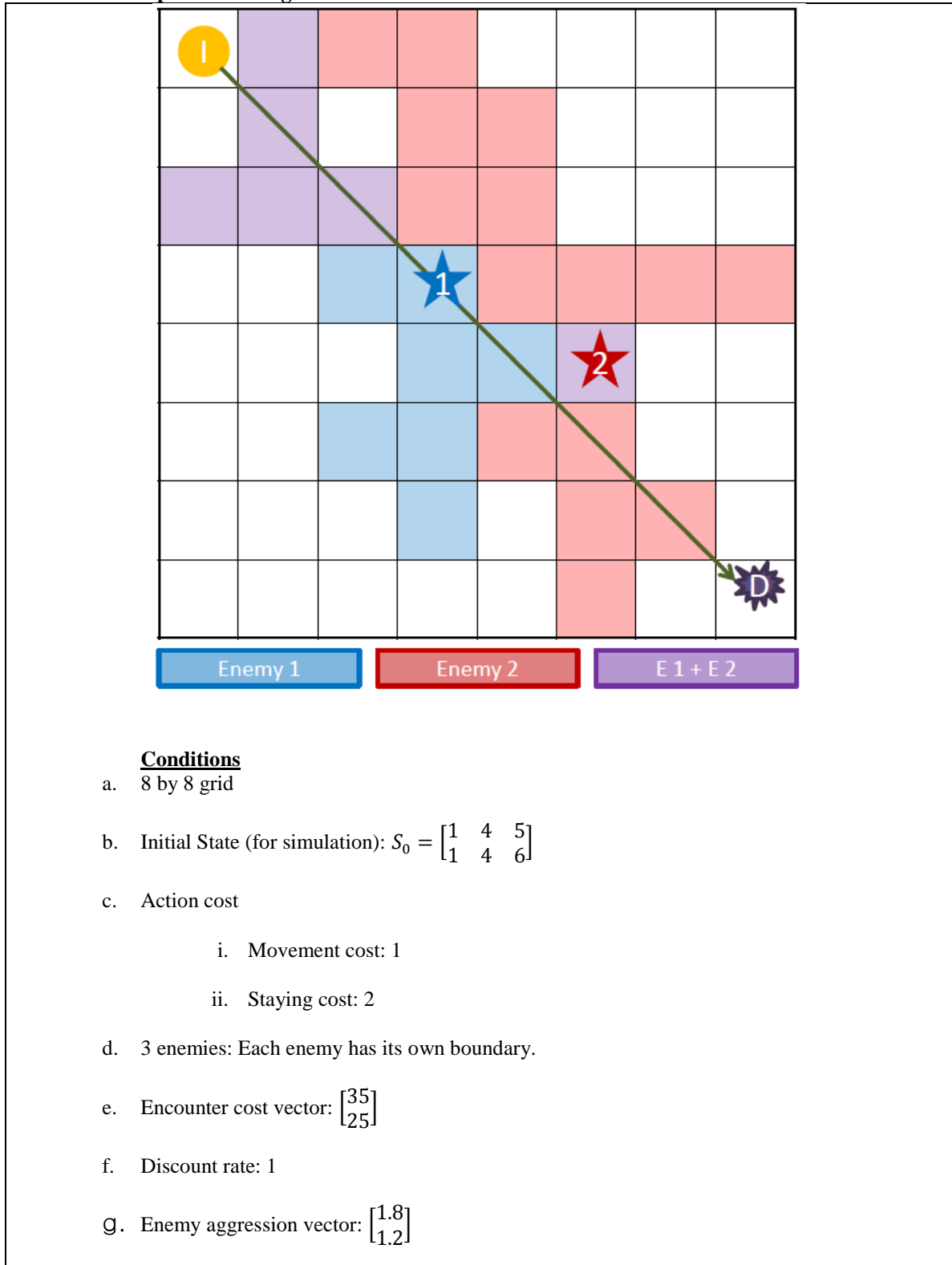
for e=1:h
    EnemyPosLoc(e).EPL=[];

    for i=1:NumRows
        for j=1:NumCols
            if HostileLoc(i,j) == 1
                EnemyPosLoc(e).EPL=[EnemyPosLoc(e).EPL,[i,j]];
            end
        end
    end
    SizeofPosELoc(e)=size(EnemyPosLoc(e).EPL,2);
end
end

```

## 6. Example of large scale problem: using approximations

### 6.1. Experiment design



### 6.2. Result of approximation 1

0	0	0	0	0	0	0	0
0	1000	187	0	0	0	0	0
0	0	813	284	0	0	0	0
0	0	0	716	340	41	0	0
0	0	0	38	624	297	41	0
0	0	0	0	38	624	297	41
0	0	0	0	0	38	624	338
0	0	0	0	0	0	38	1000

Enemy 1

Enemy 2

E1+E2

[Number of Visiting during 1000 iterations]

#### Results:

- Completion time: ~ 55 min
- average number of encounter: 0.37
- average encounter cost: 11.92
- average step size: 7.419
- average action cost: 7.419
- most frequent visiting routes

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{bmatrix}$$

### 6.3. Result of approximation 2

0	0	0	0	0	0	0	0
0	1000	530	0	0	0	0	0
0	9	492	811	0	0	0	0
0	0	1	280	794	290	0	0
0	0	0	50	350	354	290	0
0	0	0	0	46	315	369	290
0	0	0	0	6	37	300	659
0	0	0	0	0	4	41	1000

Enemy 1

Enemy 2

E1+E2

[Number of Visiting during 1000 iterations]

#### Results:

- Completion time: ~ 95 min
- average number of encounter: 0.139
- average encounter cost: 3.685
- average step size: 8.318
- average action cost: 8.318
- most frequent visiting routes

$$\begin{bmatrix} 1 & 2 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 8 \end{bmatrix}$$