A Study on Determination of Optimal Flow Paths for Safety Injection Using the Dijkstra Algorithm

Soon Ho Park, Jae Hwan Kim, Dae Seop Kim, and Man Gyun Na
Department of Nuclear Engineering, Chosun University, 309 Pilmun-daero, Dong-gu, Gwangju, Korea
*Corresponding author: magyna@chosun.ac.kr

1. Introduction

After the Fukushima nuclear power plant accident occurred by the East Japan Great Earthquake, the public's concern and interest for the safety of a nuclear power plant is growing. The cause of these concerns and interest is because of not quickly checking the status of the plant in the incident or accident situations and not appropriately responding to each situation. Therefore, it is necessary to develop safety measures for the worst conditions. Therefore, the accident recovery system is necessary to achieve the success path for stable shutdown from severe accidents.

In this study, we have analyzed the optimal flow paths for safety injection as a part of the success path analysis from the severe accident for the composition of the accident recovery system.

In order to solve the optimal flow path problem among success paths, the Dijkstra algorithm was used and the possibility of its applicability was confirmed through a simple example.

2. Determination of Optimal Flow Paths

The Floyd algorithm and Dijkstra algorithm are used to solve the shortest path finding problem. In the Floyd algorithm, if we want to know the shortest paths from one particular vertex to all the others, the Floyd algorithm would be inappropriate. Therefore, we will use the Dijkstra algorithm for the shortest path problem [1].

2.1 Basic concept of Dijkstra algorithm

The Dijkstra algorithm to solve the problem of the shortest paths is widely known as the network analysis method. In order to calculate the minimum cost (or weight) of each node, the Dijkstra algorithm has been developed.

This algorithm determines the shortest distance of each node from the starting node, and finally finds out the shortest distance of all the nodes.

We first need to specify a starting node, and can calculate the shortest distance without double-counting unless the starting node is changed.

In order to implement the algorithm, the adjacency matrix should be organized by using the ID value of the node of each loop (the adjacency matrix shows the value of the connection relation, and the values are loop diameter, length, etc.). Therefore, weight matrix is used, and nodes which are not directly connected are assigned at the maximum value of the random sets.

2.2 Main implementation steps of Dijkstra algorithm

The Dijkstra algorithm in the direction graph which does not have a negative value determines the shortest path between the destination node and the starting node. This algorithm such as Label-setting technique finds the smallest distance value among the cumulative values according to all iterations [2-4].

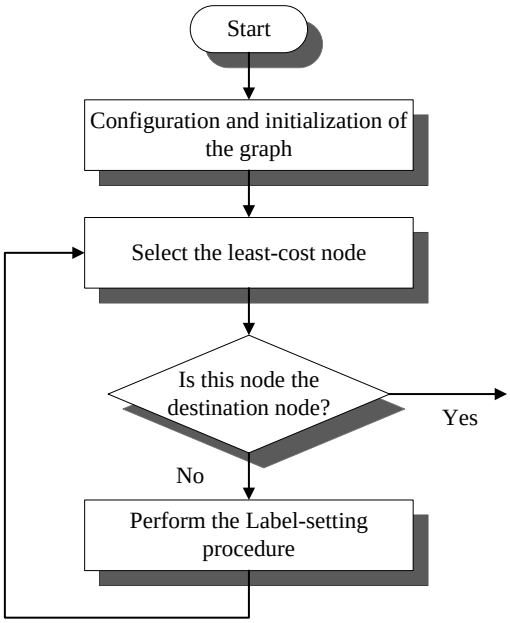

Fig. 1. Flowchart of Dijkstra algorithm

Fig 1 shows the flowchart of the Dijkstra algorithm.

In the Dijkstra algorithm, the shortest path from the starting node **X** to the other nodes is constructed in order of increasing length until the destination node **Z** is reached. In order to solve the problem, the node and branch are divided into three sets, respectively [3]. The nodes are as follows.

Set A: the nodes which the path of minimum length from **X** is known; nodes will be added to this set in order of increasing minimum path length from **X**

Set B: the nodes which the next node to be added to set **A** will be selected; this set comprises all those nodes that are connected to at least one node of **Set A**

Set C: the remaining nodes.

The branches are as follows.

Set 1: the branches occurring in the shortest paths from **X** to the nodes in **Set A**.

Set 2: the branches which the next branch to be placed in the **Set 1** will be selected; one and only one branch of this set will lead to each node in the **Set B**.

Set 3: the remaining branches (rejected or not yet considered).

To begin with, all nodes are in **Set C** and all branches are in **Set 3**. The node **X** is transferred to **Set A** and the following steps are performed repeatedly.

Step 1: consider all branches **y** connecting the node just transferred to **Set A** with a node **Y** in **Sets B** or **C**. If node **Y** belongs to **Set B**, we investigate whether the use of branch **y** gives to a shorter path from **X** to **Y** than the known path that uses the corresponding branch in **Set 2**. If this is not so, branch **y** is rejected; however if use of branch **y** results in a shorter connection between **X** and **Y** than hitherto obtained, it replaces the corresponding branch in **Set 2** and the Latter is rejected. If the node **Y** belongs to **Set C**, it is added to **Set B** and branch **y** is added to **Set 2**.

Step 2: Every node in **Set B** can be connected to **X** in only one way if we restrict ourselves to branches from **Set 1** and one from **Set 2**. The node with minimum distance from **X** is transferred from **Set B** to **Set A**, and the corresponding branch is transferred from **Set 2** to **Set 1**. We repeat the process until node **Z** is transferred to **Set A**.

And then the solution has been found.

2.3 Confirmation of the applicability

In the situation of the severe accident, if the safety injection system (SIS) does not work properly, it is necessary to use the alternative flow path.

In this study, the Dijkstra algorithm was used to determine the optimal alternative flow path, and the possibility of its applicability was confirmed through a simple example. Table I shows the determination of optimal flow paths about the example which briefly describes a real nuclear power plant. And Fig. 2 expresses the relationship of network nodes.

The simulations were performed under the condition that the source node is node 3 and the destination node is node 16. In the situation of using the active system, the paths #1 and #2 in Table I do not work properly because the paths #1 and #2 do not include active components of a motor or pump. Therefore, path #3 through the pump can work properly. And Fig. 3 shows the optimal flow path of simulation.

3. Conclusions

In this study, we have analyzed the optimal flow paths for safety injection as a part of the success path analysis from the severe accident for the composition of the accident recovery system.

The Dijkstra algorithm was used and the possibility of its applicability to the actual nuclear power plants was confirmed through an example which briefly describes a piping system of a real nuclear power plant.

As a result of simulations, if the Dijkstra algorithm is given a variety of data such as length, diameter and condition of piping systems, it will be possible to determine the optimal flow paths for safety injection due to nuclear power plant accidents.

REFERENCES

[1] Dijkstra, E. W., A Discipline of Programming, Prentice Hall, Englewood Cliffs, NJ, 1976.

[2] Hart, P. E., N. J. Nilsson, and B. Raphael, A Formal Basis for the Heuristic Determination of Minimum Cost Paths in Graphs, IEEE Trans. on Systems Science and Cybernetics, Vol.SSC-4, No.2, p.100-107, 1968

[3] Dijkstra, E. W., A Note on Two Problems in Connexion with Graphs, Numerische Mathematik, Vol.1, p.269-271, 1959.

[4] Cherkassky, B. V., A. V. Goldberg, and T. Redzik, Shortest Paths Algorithms : Theory and Experimental Evaluation, Mathematical Programming, Vol.73, No.2, p.129-174, 1996.

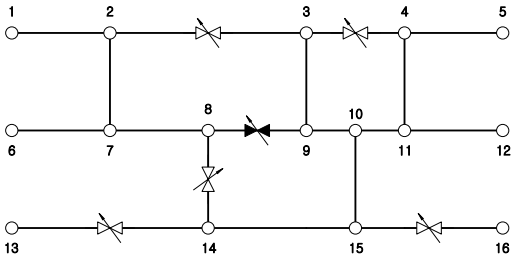

Fig. 2. Relationship of network nodes

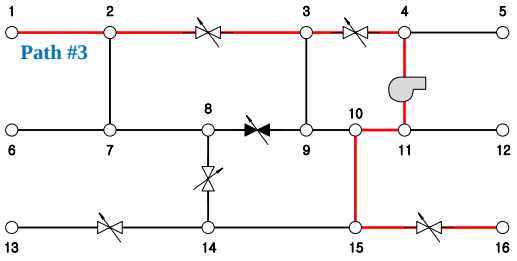

Fig. 3. Optimal flow path of simulation

Table I: Determination of optimal flow paths

(a) plow path

No	Flow path
Path #1	1 → 2 → 7 → 8 → 14 → 15 → 16
Path #2	1 → 2 → 3 → 9 → 10 → 15 → 16
Path #3	1 → 2 → 3 → 4 → 11 → 10 → 15 → 16

(b) cost, success or failure

No	Cost	Success or Failure
Path #1	7	Failure (passive path)
Path #2	7	Failure (passive path)
Path #3	8	Success