

# THE OPTIMAL CLUSTER REPAIR SEQUENCE FOR A TRANSPORTATION NETWORK FOLLOWING AN EARTHQUAKE

by

Fahad Alrakabi

---

A Dissertation Presented to the  
FACULTY OF THE GRADUATE SCHOOL  
UNIVERSITY OF SOUTHERN CALIFORNIA  
In Partial Fulfillment of the  
Requirements of the Degree  
DOCTOR OF PHILOSOPHY  
(CIVIL ENGINEERING)

December 2003

Copyright 2003

Fahad Alrakabi

UMI Number: 3133238

Copyright 2003 by  
Alrakabi, Fahad

All rights reserved.

#### INFORMATION TO USERS

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleed-through, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

**UMI<sup>®</sup>**

---

UMI Microform 3133238

Copyright 2004 by ProQuest Information and Learning Company.

All rights reserved. This microform edition is protected against  
unauthorized copying under Title 17, United States Code.

ProQuest Information and Learning Company  
300 North Zeeb Road  
P.O. Box 1346  
Ann Arbor, MI 48106-1346

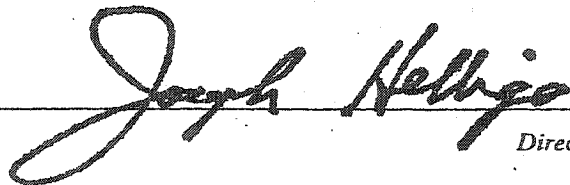
UNIVERSITY OF SOUTHERN CALIFORNIA  
THE GRADUATE SCHOOL  
UNIVERSITY PARK  
LOS ANGELES, CALIFORNIA 90089-1695

*This dissertation, written by*

Fahad Alrukaibi



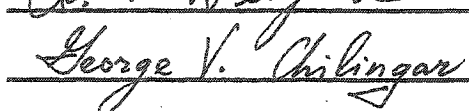

*under the direction of his dissertation committee, and  
approved by all its members, has been presented to and  
accepted by the Director of Graduate and Professional  
Programs, in partial fulfillment of the requirements for the  
degree of*

*DOCTOR OF PHILOSOPHY*

  
Director

*Date* December 17, 2003

*Dissertation Committee*

	<u>James E. Moore, II</u>
	Chair
	<u>L. Carter Wellford</u>
	<u>George V. Chilingar</u>
	<u>Harry W. Richardson</u>

## Table of Contents

List of Tables.....	iv
List of Figures.....	v
Abstract.....	vi
1. Introduction.....	1
1.1 Rationale.....	1
1.2 Network Performance.....	2
1.3 Problem Statement.....	3
1.4 Dissertation Layout.....	6
2. Literature Review.....	7
2.1 Traffic Assignment.....	7
2.2 Bridge Prioritization Models.....	9
2.3 Effect of Earthquakes in Transportation Networks.....	13
2.4 Post – Earthquake Transportation Network Performance.....	15
2.5 Dynamic Programming.....	18
2.6 Dynamic Programming (DP) and Vehicle Routing Problem (VRP)....	21
2.7 Impact of Earthquakes in the Economic Sector.....	24
3. Methodology.....	27
3.1 Efficiency in Computing of Sequence Optimization Problem.....	30
3.2 Mathematical Interpretation.....	33
3.3 Method I: A Dynamic Programming Approach to the Modified TSP for the Cluster Sequencing Problem.....	37
3.3.1 Dynamic Programming Formulation.....	42
3.3.2 Number of possible cluster repair sequences formulation.....	47
3.4 Method II: Shortest Path Algorithm for the Cluster-Sequencing problem.....	54

3.4.1 Cluster-sequencing problem as a shortest path problem.....	55
3.4.2 Pseudo-code for Shortest Path Algorithm Used for the Cluster- Sequencing Problem.....	59
3.4.3 The Total Delay in the Transportation Network.....	62
3.5 Method III: The Heuristic Approach for the Cluster-Sequencing Problem.....	64
3.5.1 Cluster-sequencing problem using the heuristic approach the “Pseudo-code”.....	66
3.6 Binary Numbers representing status for cluster repair.....	68
4. Examples and Analysis of Cluster-Sequencing Problem Solution Methods.....	70
4.1 Applying the cluster-sequencing algorithms in a specific example....	71
4.1.1 Examples for the shortest path and heuristic approach algorithms...	71
4.2 Results of the traffic assignment from the study area.....	80
4.3 Testing Network Configurations Results by a Simulating method....	81
4.3.1 Examples from the simulation method.....	83
4.3.2 Testing different number of clusters.....	87
5. Conclusion and Future Work.....	90
References:.....	95
Appendix A: Area of Study.....	99
Appendix B: TransCAD Results:.....	104
Appendix C: C++ Codes.....	107
Appendix D: Codes for testing the simulating method.....	115

## List of Tables

Table 1: Mode Share in Commute trips.....	15
Table 2: Values of the states for 3 clusters .....	72
Table 3: States represented in binary numbers for 3 clusters to indicate clusters conditions .....	74
Table 4: Values of the states for 4 clusters .....	74
Table 5: States represented in binary numbers for 4 clusters to indicate clusters conditions.....	78
Table 6: Simulation results for 3 clusters.....	83
Table 7: Different results for the simulating method in 5 clusters.....	84
Table 8: The results of each algorithm for different runs.....	85
Table 9: Average results from different number of clusters.....	88
Table 10: Closed links number for each cluster.....	100

## List of Figures

Figure 1: One Stage Analysis.....	19
Figure 2: Multistage Analysis.....	20
Figure 3: The Proposed Solution Methods Approaches. ....	32
Figure 4: Four clusters plus start node are connected with explanatory links to show status of repairs.....	41
Figure 5: The different choices of repair sequence for 4 clusters .....	51
Figure 6: The network used for different choices of repair sequence for 4 clusters.....	52
Figure 7: Number of states for 4 Clusters .....	57
Figure 8: Binary Numbers represent the different cluster repair sequences .....	69
Figure 9: States as numbers for 3 clusters and results from the algorithms .....	75
Figure 10: Shortest path method result in dash arrows, and heuristic approach algorithm results in bold arrows.....	79
Figure 11: The differences between the applied algorithms .....	86
Figure 12: Different number of clusters for several code tests .....	89
Figure 13: San Francisco Bay Area transportation network .....	101
Figure 14: Bay Area clusters .....	102
Figure 15: The specified area of study in the Bay Area.....	103

## Abstract

After a major earthquake bridges and other network links will most likely have to be grouped into clusters to define a repair sequence. This will result in a finite number of clusters. We are concerned with the method for sequencing repair of these clusters, given that resources may not be available to repair all clusters simultaneously. The optimal repair sequence minimizes total network delay. Our aim, therefore, is to develop methods that can optimize network performance as incremental repairs are made, until there is no further damage to the transportation network. We have established different solutions methods to handle the cluster-sequencing problem. As a way to track a solution method for our problem, there is a formulation of a dynamic programming approach to the modified travel salesman problem. Then the solution methods proposed are the Dijkstra's shortest-path algorithm, and a heuristic greedy approach. There is also a newly developed simulation method that is applied to the cluster-sequencing problem.



## **1. Introduction**

### **1.1 Rationale**

Transportation networks are part of the infrastructure system of any urban area. Bridges and road networks constitute a major part of the transportation system, and their construction requires a large investment. After an earthquake, bridges and road networks need to be repaired. Bridges are the links in the transportation networks that are most likely to be damaged during an earthquake. Because bridges are considered critical components in transportation systems, many bridges are closed following an earthquake. Obviously, damage to bridges will have a deleterious effect on the performance of a transportation network.

Inspection or repair of bridges is one of the logistical problems associated with earthquakes. Depending on whether or not an earthquake occurs, pre-earthquake maintenance and post-earthquake repair must be considered. In the pre-earthquake scenario, maintenance or inspection of bridges is usually considered in order that the seismic retrofit and upgrade meet design codes requirements. Pre-earthquake maintenance in general deals more with the structure and design of the bridge than with transportation network problems. There is a need for an efficient repair method because post-earthquake repair leads to a severe congestion of the transportation network. It is a result of that many links is closed at time of the repair.

## 1.2 Network Performance

In this research we are concerned with post-earthquake repair of bridges. The methods of repair that will be proposed in the research outline below focus on the transportation network and its improvement. The goal is to develop methods that can optimize network performance as incremental repairs are made, until there is no remaining damage to the transportation network.

Maintenance of bridges after an earthquake, which includes inspection and repair, needs to take into account time and financial resources. As a result, we should allocate available resources according to optimization theory. Authorities need to relieve congestion resulting from damaged bridges and links that have been closed. Therefore, there will be a reduction in travel costs incurred by drivers. Previous research shows that damage to bridges can be significantly effective, as repair process is time consuming and also number of paths is generally less in the network (Werner and Cooper, 1991).

Methodology is needed to rehabilitate these closed links and to repair damaged bridges. The goal of this methodology should be to optimize network performance. Transportation network performance can be assessed by travel time, road safety, accessibility, network delay and other measurements. Network reliability could be one of the potential measures of network performance.

Network performance in this research will be measured by the reduction of time delay in the network. There is much need for this kind of assessment. Improving network performance provides an estimate for how much is saved in terms of travel costs.

The research problem is a multistage optimization problem with a finite number of stages. Each stage includes a different number of states. Sequence of time periods is needed to repair links and bridges. We plan to optimize the order of repair. Application of this method can reduce the total delay resulting for a given number of closed bridges and segments in the transportation system.

### **1.3 Problem Statement**

After a major earthquake, bridges and other network links will most likely have to be grouped into clusters in to define a repair sequence. This will result in a finite number of clusters. We are concerned with the method for sequencing repair of these clusters, given that resources may not be available to repair all clusters simultaneously. Our goal is to determine the sequence in which the clusters should be repaired. The optimal repair sequence minimizes total network delay. The grouping or clustering of bridges can be treated as a separate part of the problem.

We have established different solutions methods to handle the cluster-sequencing problem. As a way to track a solution method for our problem, there

is a formulation of a dynamic programming approach to the modified travel salesman problem. Then the solution methods proposed are Dijkstra's shortest-path algorithm, and a heuristic greedy approach. There are sections specified for each of the solution methods. In each section, there will be an explanation for how each solution method can handle the cluster-sequencing problem. There will be a test for each approach and a comparison of the solution procedures each method uses. Advantages and difficulties for each method are also discussed.

There is also a newly developed procedure that will be applied to the cluster-sequencing problem. The method, an application of a simulation, examines the different possible network configurations and clusters. The simulation method builds the states network for any number of clusters.

Binary numbers are applied to the cluster-sequencing problem. The different states in the transportation network, which represent the different cluster sequences, are explained by using a binary numbers concept. The idea is to show a different 0 or 1 vector representing each different cluster repair sequence.

In general, mathematical models are applied to solve the problem of cluster-sequencing. Previous studies in the bridge-sequencing field were concerned only with the sum of weighted bridge structural and transportation characteristics, and these approaches did not use a multistage method as it will be shown in the literature review section. Our method, on the other hand, scores the cluster, which includes bridges depending on degree of benefit to the

transportation network system. Therefore, cluster that improves the network performance for the whole transportation network system more is repaired first.

Our problem is unique not only because of the focus on sequencing-clusters, but also because of the different methods we utilize to reach a solution. Therefore, our goal is to show that the solution to the cluster-sequencing problem is generally approachable with different methods. Then analysis techniques are applied between the solution methods.

In general, all the prior studies and literature seek to optimize some parts or paths within transportation networks. Our approach is to deal with the results obtained for the whole transportation network system. Furthermore, we are looking to improve the old methodologies used in the reconstruction of damaged bridges. Ideally, in the end, overall network performance will be improved.

There is an important use of TransCAD, a leading transportation planning software used to calculate the result of the different transportation network configurations. Here, TransCAD is used to apply the traffic assignment procedure for the study area.

In the study area, bridges are clustered depending on certain variables. Bridge damage is scaled based on the bridge damage index (BDI), which is the degree of damage to a bridge, the result of a separate research project.

Detailed examples will be given, explaining the procedure used in sequencing the clusters. Furthermore, in the research outline some important definitions will be explained.

#### **1.4 Dissertation Layout**

The next chapter presents a comprehensive academic review of the relevant literature. We will review the literature with respect to how important it is to deal with emergency situations from the transportation standpoint. The literature review is arranged to show the main ideas that explain the related of this research problem. It starts with the importance of the traffic assignment in transportation field. The section continues with the bridge prioritization models. Next, section 2 demonstrates also some other important topics related to the cluster-sequencing problem.

Section 3 deals with the methodology, and is divided into six sub-sections, each of which will give an example and explain the methods of solution. The methodology includes details of the solution methods procedures. It explains how the cluster-sequencing problem can be approached from different methods.

Analysis and comparison between the applied methods will be discussed in section 4. It uses a simulation method for the purpose of results comparison. Section 5 consists of conclusions and identifies future work.

## 2. Literature Review

### 2.1 Traffic Assignment

The aim with respect to traffic assignment is to determine types of movements satisfying travel demand, compatible with the transportation network and drivers characteristics (Thomas, 1991). All route choices may be represented by travel cost. The measure of cost used depends on the transportation characteristics and dominant traffic conditions. The traffic assignment problem can be solved when there is a given O-D trip matrix representing the demand for trips. If all drivers have full and complete perception of costs on the network, then the resulting state of equilibrium is referred to as deterministic. When drivers differ in their perceptions of costs, the state of equilibrium is referred to as stochastic.

User-equilibrium means that all travelers will choose routes that minimize their travel times. User equilibrium will be found when travelers cannot improve their travel times by changing to any other route (Doohee, et al., 1999). When user equilibrium is achieved, the time of each route is the shortest possible within the transportation system. As the degree of congestion in a network increases, the

percent of modeling error tends to increase, and the calculations of user equilibrium conditions will change in accuracy (Sheffi, 1985).

Almost all current traffic assignment models assume that traffic flow is equally distributed at time of assignment (Thomas, 1991). However, as pointed out by Ben-Akiva (1985), in periods of high demand observed traffic flows are not stable.

According to Thomas (1991) the basic goal of the traffic assignment process is to regenerate the type of traffic flow in the transportation network. In addition, the two major goals of traffic assignment procedures can be summarized as:

1. To evaluate the traffic volume on the links of the transportation network.
2. To estimate cost of travel from origins to destinations.

According to Sheffi (1984), Patriksson (1994), and Thomas (1991) traffic assignment procedures can be used to address a number of questions:

- Traffic assignments results can help improve constructional priorities by assigning the expected trips at several points.
- If the traffic flow is assigned to some specified paths what are the advantages and disadvantages for the transportation system?
- Measuring the weakness in the present transportation system by assigning the present origin-destination matrix to the system. As a result, bottlenecks in the transportation system will be highlighted.



- Determining the path used between each Origin-Destination node.

Each used path will have the same traveling time. The links used to establish the path can therefore be determined.

## **2.2 Bridge Prioritization Models**

Bridge prioritization for repair is the major topic of the proposed research. Most of the literature available in related transportation research focuses primarily on the retrofitting of bridges before an earthquake. Minimal work has been done on the post-earthquake prioritization of bridges. The focus of this research is post-earthquake prioritization for repair.

In the seismic retrofitting field Basoz and Kiremidjian (1995) describe the seismic retrofitting of bridges. They mention how Geographic Information Systems (GIS) can be used in this field. A Geographic Information System based prioritization method is developed. Retrofitting bridges under limited resources requires a method for establishing the order of the work. This method ranks bridges depending on the vulnerability and importance of each bridge. The most critical bridge is defined as a bridge that would prevent accessibility of the network. Bridges with the same seismic characteristics behavior are grouped. It is very expensive and time consuming to retrofit all bridges.

A detailed review of the available prioritization approaches is presented in Basos and Kiremidjian (1994). The main prioritization approaches are the Caltrans approach, ATC approach, Illinois department of transportation approach, Washington state department of transportation approach, as well as other approaches. The report also mentions the limitations of these existing approaches. A new prioritization method has recently been developed to identify high-risk bridges and prepare them for seismic retrofitting. The prioritization method has been created depending on vulnerability and importance of the bridge. The importance of a bridge is connected to its function within the transportation network system. The developed methodology considers seismic activity as the major hazard to bridges. These methods can be applied to other natural hazardous as well, such as winds or floods. Some lifeline systems other than highway bridges can be considered in this method, for example railway bridges or any important component of the lifeline networks.

#### A. California Department of Transportation (Caltrans) approach

The Caltrans approach retrofits structures that have a high degree of risk and have the greatest effect on transportation systems. The main goal of this prioritization approach is to specify the structures that can mostly be affected during an earthquake (Sheng and Gilbert, 1991). The attributes of the risk algorithm used include such information as route type, year of construction and

traffic exposure (average daily traffic). The developed algorithm accounts for a range in risk variation number, between 0 and 1. Whenever the risk number is close to 0 it means there is a relatively low level of risk. Close to 1 means a high level of risk. The risk number required for measurement is equal to the sum of each weighted, for example, structure attribute. Caltrans has also developed a long-term risk algorithm that relies on a multiplication of three weighted scores. These are Seismicity (load factor), Importance (social factor), and Vulnerability (structural factor) (Basos and Kiremidjian, 1994).

#### B. Applied Technology Council (ATC) approach

There are some limitations to the types of bridges that can be used in this approach. The first step in seismic retrofitting under the ATC approval is a preliminary screening. This requires arrange bridges according to their need for seismic retrofitting. The final decision also depends on issues other than engineering, such as political, social, and economic issues. However, the seismic rating system is created to rate the bridges only according to engineering points. There are three main variables used in rating bridges: seismicity of the bridge location, vulnerability and importance of the bridge (Basos and Kiremidjian, 1994).

### C. Illinois Department of Transportation (IDOT) approach

IDOT ranks bridges depending (as the previous methods) on the seismic risk of bridges. The seismic risk is explained by two variables: the probability of failure of a bridge, and the consequences of such a failure. To achieve the best evaluation of seismic risk for a bridge, a two-stage approach is used. The first stage is a screening stage, which gives a preliminary rank to all bridges. The second stage ranks bridges based on a more detailed evaluation (Basos and Kiremidjian, 1994).

### D. Washington State Department of Transportation (WSDOT) approach

The WSDOT approach considers the cost estimates of the seismic risk for bridges in Washington. The major objectives of such a retrofitting procedure are to decrease the degree of risk that a bridge will collapse, and to let important bridges remain in good condition (Basos and Kiremidjian, 1994).

The reliability of retrofit methods of highway bridges is considered in many articles, including one of the most reliable from Cherng and Wen (1994). There is a method established for calculating the retrofit priority and upgrading of highway bridges. The retrofit priority is measured by the performance index PI, which is scaled from 0 to 10. The goal is to maximize the net retrofit benefit for a given budget and target network reliability. The minimum travel time between any two nodes in addition the network capacity, which is measured in vehicles

per hour, are used as the most important measure of transportation network performance.

Bonyuet (1983) proposes a methodology for road rehabilitation and bridge replacement. For a specified budget, the problem is to determine which road should be repaired and which bridges should be replaced to minimize total user costs within the budget limit. The model is formulated as a mixed non-linear programming problem with linear constraints. The method assumes that when the road investment variables are fixed, the problem becomes one of linear programming. The model is developed to aid decision makers in the allocation of limited funds in the improvement of large-scale road-bridge transportation systems. The solution methodology includes a wide search for integer variables and a trial and error procedure between the traffic assignment problem and the road rehabilitation budget allocation problem.

### **2.3 Effect of Earthquakes in Transportation Networks**

This section will show how earthquakes are responsible for damages and delay in transportation networks, focusing on an earthquake that occurred in the United States. The Loma Prieta California earthquake was chosen as an example to illustrate how earthquakes can affect transportation infrastructure. Possible loss

of life and the effect of an earthquake on the economy will be covered separately in a later section.

The Loma Prieta earthquake occurred October 17, 1989 measuring 7.0 on the Richter scale. It caused extensive damage to San Francisco Bay Area highways. Caltrans (1994) indicates that damage occurred to 91 state highways bridges, thirteen of which were closed due to severe damage. One of the major impacts of the Loma Prieta earthquake was the closure of the San Francisco – Oakland Bay Bridge. Kameda, et al (1990) mention that the period of reconstruction of the Bay Bridge, approximately one month, caused an increase in traffic volume and travel time on other routes. It is a result; there were no alternative routes for most drivers. The estimated total travel time increase is estimated at 1.5 times. Other damage was also reported, including a collapsed three - quarter mile section of the Nimitz freeway in Oakland.

Deakin (1991) indicates from 1989 survey that most commuters said they tried several different routes to get to work. Furthermore, one of the findings of the survey is that drivers tried different types of transit and most of the drivers quickly selected either the Bay Area Rapid Transit (BART) system or one of the ferry services.

The survey from Deakin (1991) indicates different mode shares between the before and after earthquake scenarios as shown below in Table 1.

	Before %	After %
Drive alone	37	10
Shared ride	24	1
Bus	10	1
BART	35	75
Ferry	--	10
Other	8	3

**Table1:** Mode Share in Commute trips

A summary of the survey results indicates that 69% of all respondents faced difficulties in their trips. As a result, most of them switched to other modes of transport, such as BART.

## 2.4 Post – Earthquake Transportation Network Performance

Post-earthquake transportation network performance is discussed in this section. We review different earthquakes that occurred in recent years in the United States and Japan. Transportation network performance is defined as a

measure in the accessibility of transport, or delay in travel time between any two nodes in the network or in the transport system as a whole. In general, earthquakes such 1994 Northridge earthquake, 1995 Kobe earthquake and others, substantially affect the performance of the transport system. Chang and Nojima (1998) indicate that one of the effects of earthquakes studied show that transportation system may lose 30 to 55 percent of pre-earthquake traffic volume. As a result, Chang and Nojima (1999) developed a measure of transport accessibility (network performance) that they applied to Kobe city in Japan. The method depends on measuring the total length of network that is open (L), and total distance accessibility (D). Each variable is estimated as a ratio between the post-earthquake and pre-earthquake situation. The developed measure of transport accessibility ranges the values from 0 to 1 and means that the transport system is completely damaged or not damaged, respectively. Measuring network performance is important for several different reasons. From Chang and Nojima (1999) described the following:

- Enable more analysis of different types of earthquake in different points.
- Scaling repair improvement through reconstruction progress.
- Establishing a good post-earthquake reconstruction plan.
- Estimating economic impact.



Wakabayashi and Kameda (1992) evaluated transportation network performance after the Loma Prieta earthquake. They introduced a method that described and accounted for subsequent traffic conditions. From the results obtained, they were able to make an estimation of network performance. Finally, network performance was tested in the case of the Loma Prieta earthquake. There was an assumption regarding the origin – destination (OD) data for the network, two cases of which are important to mention with respect to our study. The two different cases of OD for post – earthquake that Wakabayashi and Kameda (1992) use are:

- Assumption 1: The network is loaded (after the earthquake) with the OD matrix from before the earthquake. This assumption is used if the traffic demand before the earthquake is still the same without changes.
- Assumption 2: Using OD matrix for the case after the earthquake is not important because of the difficulty in obtaining an exact estimate of the OD matrix. This assumption is used based on the number of people who switched to BART after the earthquake. In addition, by knowing the average number of occupancy per vehicle, it will be feasible to estimate the OD matrix following an earthquake.

It is extremely important that transportation network performance is accurately estimated and evaluated in order to achieve a reliable network, which is connected and accessible under all circumstances.

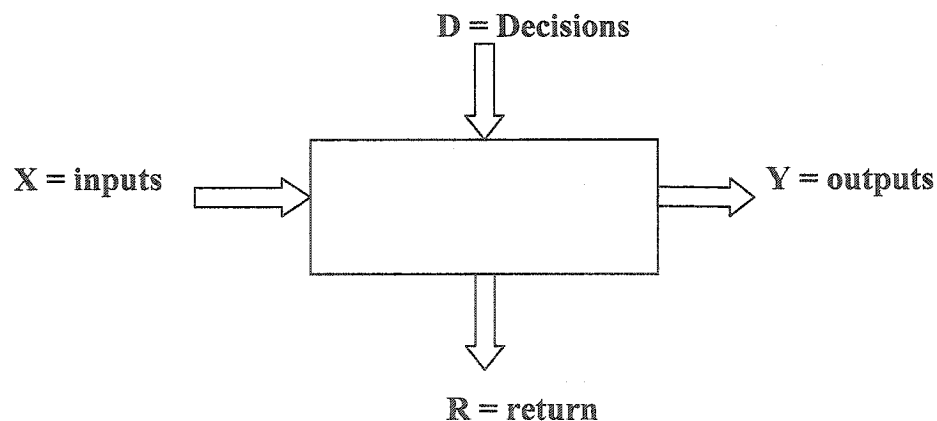
## **2.5 Dynamic Programming**

This part of the dissertation is included to describe dynamic programming in detail. The development of dynamic programming is largely due to Richard Bellman and his colleagues, Dynamic programming is one of the important optimization techniques. It is usually used in the operations research field and others to solve difficult optimization problems. In general, dynamic programming obtains most solutions by working backwards. Nemhauser (1966) emphasizes that a good approach of dynamic programming is to simplify the variables and decisions used before applying optimization. This will lead to the same optimization goal but with an easier model. Dynamic programming is based on the principle of optimal operation at every step of its application. The principle of optimality was developed by the founder of dynamic programming: Bellman said:

*“An optimal set of decisions has the property that whatever the first decision is, the remaining decisions must be optimal with respect to the outcome which results from the first decision” (Bellman, 1957).*

Dynamic programming depends on a sequential or multistage decisions procedure. Results from dynamic programming are accomplished by adding the total output of all sequences or stages.

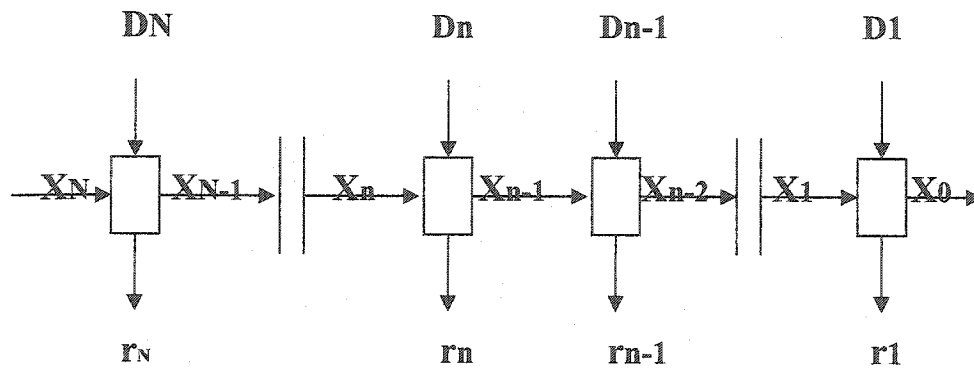
Nemhauser, (1966), explains the multistage process in dynamic programming in the following two flowcharts. Figure 1 is an illustration of a one-stage process. It shows how the four variables: inputs, decisions, return, and output can determine analysis in each stage.



**Figure 1:** One Stage Analysis

Figure 2 shows how the complete set of processes in a dynamic program interacts. At each different stage there is a dependency on the previous stage.

The output from one stage will be an input to next stage until the last stage is achieved.



**Figure 2:** Multistage Analysis

Kiyota, et al (1999) proposed a multistage dynamic programming method. The method determines the impact of construction in transportation networks. This is done by discussing three important issues. These are the optimal groups of road segments to be repaired, and the sequence of construction and closure strategy of links. The method assumes that optimal solution is obtainable by maximizing the sum of driver utilities over a finite period by using dynamic programming. After applying the constraints concerns about budget, the method suggest this formulation of the problem:

$$V_t(\mathbf{X}) = U_t(\mathbf{Z}, \mathbf{P}) \quad (k = 1)$$

$$V_k(\mathbf{X}) = \max_{y \in Y_{k-1}} [V_{k-1}(\mathbf{y}) + U_k(\mathbf{Z}, \mathbf{P})] \quad (k = 2 \sim K)$$

Where  $V$  is the cumulative utility,  $U_k$  is the utility over all links, vector  $\mathbf{X}$  is a binary variable to denote the status of construction, vector  $\mathbf{y}$  denotes the state of links at the  $(k - 1)$ th period, vector  $\mathbf{P}$  denotes the usage pattern, and vector  $\mathbf{Z}$  to denote the progress of construction as  $\mathbf{Z} = 2\mathbf{Y} - \mathbf{X}$ .

Winston, (1994), summarizes the general characteristics of dynamic programming applications as:

1. **Stages:** each solvable problem by dynamic programming consists of several stages.
2. **States:** each stage consists of several states.
3. **Decisions:** a translation of the transformation from current state to the next state at next stage.
4. **Principle of Optimality:** the problem should satisfy the principle of optimality explained above.

## 2.6 Dynamic Programming (DP) and Vehicle Routing Problem (VRP)

The first studies of the vehicle routing problem were discussed by Dantzig and Ramser (1959) more than 40 years ago, when they applied a linear

programming procedure. However, the Vehicle Routing Problem (VRP) consists of a central node or a depot. It requires finding a set of least cost vehicle routes by visiting all nodes (customers) in the network. It starts and ends at the depot. The VRP is subject to various constraints such as vehicle capacity or route length restrictions. The VRP is known to be difficult to solve exactly and heuristic approaches are usually employed. The main objectives of the VRP are summarized by Toth and Vigo (2002). The most important objectives are to minimize the whole transportation cost (distance of travel), and to balance the routes for travel time and vehicle load.

The Traveling Salesman Problem (TSP) is one of the best-known routing problems. In TSP a number of cities have to be visited by a salesman who must return to the same city where he started. The solution of the problem is to find the route that minimizes the total distance traveled.

There are many benefits to combining solutions of the vehicle routing problem by using dynamic programming in comparison to other methods. Psaraftis (1978) indicates that some features of dynamic programming make it more suitable to handle vehicle routing problems such TSP. The most important feature of using dynamic programming is generality of handling many types of objectives functions. There is only one important condition for an objective function to be handled as dynamic programming, and that is separation capability. The function will be separable if it is possible to differentiate between

current stage and last stage times. It is known that other than dynamic programming methods can handle more cluster/nodes in solving TSP, but they cannot deal with any change in the shape of objective function. The other feature mentioned in the article is the relative ease of code dynamic programming in the computer.

The first of the papers to deal with dynamic programming and vehicle routing problems (especially TSP) are by Bellman (1962) and Held and Karp (1962). Two years later, Clarke and Wright (1964) introduced a greedy algorithm to the algorithm of Dantzig and Ramser (1959). Malandraki and Dial (1996) introduced a heuristic algorithm for the time dependent traveling salesman problem. The restricted DP heuristic algorithm is a general form of the nearest neighbor heuristic but results in better solutions. The solution method is therefore considered to be in the middle between the optimal solution and the nearest neighbor method.

Furthermore, there is a substantial literature dealing with VRP. The studies are as varied as the different types of VRP. Toth and Vigo (2002) summarize the different types of VRP problems as capacitated and distance constraint, VRP with backhauls, VRP with time windows, and VRP with pickup and delivery.

In most cases there was no concern in the literature with linking dynamic programming and VRP problems. In addition, as mentioned before, there are

many benefits to combining dynamic programming and VRP. Our proposal therefore links these two fields. Moreover, it is necessary to handle and improve all the changes and assumptions, specific to our problem, differing from standard methods.

## **2.7 Impact of Earthquakes in the Economic Sector**

Earthquakes cause severe damages to transportation network systems and the infrastructure of all other lifeline systems. Besides the direct damages of the earthquakes there are indirect effects such as traffic congestion and economic repercussions. Damages to transportation networks are one of the main causes of the reduction in the production output, which leads to a reduction in the economy.

Tomofumi and Hideki (1997) estimate the socio-economic effect of earthquake damage on transportation facilities by combining three methods: transportation state estimation after an earthquake, transformation of the result of traffic estimation to change of production and economic effect estimation.

Nigg (1996) conducted a study using random sample of businesses in Memphis, Tennessee to show how lifeline systems damages can affect many types of businesses. The results of the study indicate how sensitive businesses are to economic disruption due to lifeline system failures.



Cho, et al, (2000) integrated bridge structure performance models; transportation networks models, system input-output models, and spatial allocation models. The goal is to model the effects of an earthquake on industrial output and transportation networks supply and demand. Furthermore, results from the integrated model are used to analyze different bridge reconstruction scenarios. Bridge type and earthquake place are the main variables to the use of fragility curves (Shinozuka, 1999). They are used in the integrated model to give the probability distribution of bridge damage conditions. Bridge damage is defined by the bridge damage index varying from 0 to 1, which means no damage and collapse respectively. The full cost of the earthquake was estimated to be around \$93.5 billion and the total travel cost with bridge repair cost was estimated to be \$1.72 billion around 1.75% of total. The other loss cost of damages was due to structure loss, and business loss.

Damages occurring to lifeline systems are the known result from earthquakes. More importantly, there is an associated loss of life, for example, the Loma Prieta earthquake caused 63 deaths, and 13,757 injuries (Loma Prieta Earthquake, San Francisco, California October 17, 1989. (n.d.). Retrieved July 11, 2002, from [http://nisee.berkeley.edu/loma\\_prieta/loma\\_prieta.html](http://nisee.berkeley.edu/loma_prieta/loma_prieta.html)). Also the Kobe earthquake caused huge loss of human life and thousands of injuries.

Chang and Nojima (1997) developed a performance measure for transportation networks that is good for estimating the economic impact. They

define economic impact as the loss of economic production output standards.

Total traffic volume is used as a measure of economic loss

$$(Z) = k \sum_{t=1}^T (1 - V_t)$$

Where  $k$  = constant,  $t$  = time index,  $T$  = time to complete highway restoration,  
and  $V$  = the ratio of post-earthquake traffic volume to pre-earthquake volume.

### 3. Methodology

Due to limited resources, damaged bridges need to be repaired in order of priority. Sequencing rules are applied to existing clusters of bridges. Clustering the bridges into repair projects is a separate problem in itself. In general, we carried out this study based on the assumption that the clustering problem has already been solved. But there is a trial for clustering the bridges based on some bridge factors. In each time, the cluster numbers are fixed. The bridge factors can be the bridge damage index (BDI) or position of the bridge.

It should be noted that the problem structures and solution approaches in the following sections describe the cluster-sequencing problem and try to find a method of solution. The clusters are a group of different damaged bridges. Our problem—solving the sequence of bridge clusters—is a unique one. Also, the cluster-sequencing problem solution is approached by different optimization methods. Furthermore, all the previous studies and literature, mentioned in the last sections, seek to optimize some parts or paths of the transportation networks, whereas our approach is to deal with the results obtained from the whole transportation network system. Our goal is to show that the solution to the cluster-sequencing problem is generally approachable from different optimization methods. There is a formulation of the cluster –sequencing problem by a dynamic programming approach to the modified traveling salesman method. The solution

methods proposed are Dijkstra's shortest-path algorithm, and a heuristic greedy approach. Each solution method technique has its relative advantages.

The traffic assignment step determines the amount of traffic that will use the various routes between origin and destination nodes in the transportation network. As a result, links will be loaded differentially depending on the origins and destinations as well as on the traffic conditions on each link. The problem addressed in this research deals with clusters that include some earthquake-damaged bridges. As a result, links connected to these damaged bridges will either be closed or have some partial reduction in capacity. Some links or network segments may therefore receive most of the generated traffic, while others may receive no additional traffic. The measures of link performance can be used to estimate the degree level of service and to suggest cause of capacity problems (Papacostas and Prevedouros, 1993).

The research goal is to calculate network performance (system cost or time delay). Traffic volume and travel time depend on the network configuration, which is decided by the repair sequence.

Within the modeling concept of transportation network analysis, traffic volume and, hence, travel time and total system cost of a specific network configuration is different from other network configurations. Travel demand may follow minimum travel time paths. If certain links are closed, then the minimum

path should be different from the baseline or other network configurations. That is why each network configuration has different link volume and travel time.

As a way to handle the cluster-sequencing problem, we establish different solution methods. The first approach is an exact algorithm and the second solution method is a heuristic approach. There are sections specified for each of the solution methods. In each section, there will be an explanation for how each solution method can handle the cluster-sequencing problem. As a result, there will be a test for each approach and a comparison of the solution procedures different methods use. The methods' benefits and detriments are also discussed.

We will mention one of the main differences between our formulation of a dynamic programming approach to the modified TSP algorithm and solution method by Dijkstra algorithm for the shortest path algorithm. The difference between them, other than the completely different solution techniques, is how each one shows and figures out the transportation network graph used to solve the problem. For example, in the dynamic programming approach to the modified TSP we will deal with cluster numbers to model the problem, not with which cluster is repaired, as is the case of shortest-path algorithm. As a result, the transportation network figure shows clusters as nodes. Conversely, for the shortest-path algorithm, repaired clusters, not cluster numbers, are the nodes in the transportation network figure.

### 3.1 Efficiency in Computing of Sequence Optimization Problem

The cluster-sequencing problem is solved by different approaches. The purpose is to show how much calculation efforts can vary between one optimization method and another. Depending on the solution method used, the problem varies in its calculation time. The optimization methods applied for the cluster-sequencing problem have a variation in their time complexity. The calculation time varies as an exponential, polynomial, and linear time. The state space calculation will be explained in details by examples.

The first algorithm in the methodology section is the formulation of a dynamic programming approach to the modified TSP for cluster-sequencing problem. This approach is useful to track a solution method for the cluster-sequencing problem. It is a combinatorial optimization problem that is categorized as NP-complete. NP-complete means that number of arithmetic steps needed to solve the problem cannot be shown to be a polynomial function of the number of inputs. It is therefore clear that there is no known polynomial time algorithm for all NP-complete problems.

The second algorithm will be the Dijkstra algorithm for the shortest path. It is our first solution method for the cluster-sequencing problem. The executing time for the solution method resulting from this algorithm will be polynomial.

There will be more details about this algorithm and its features in the specified section.

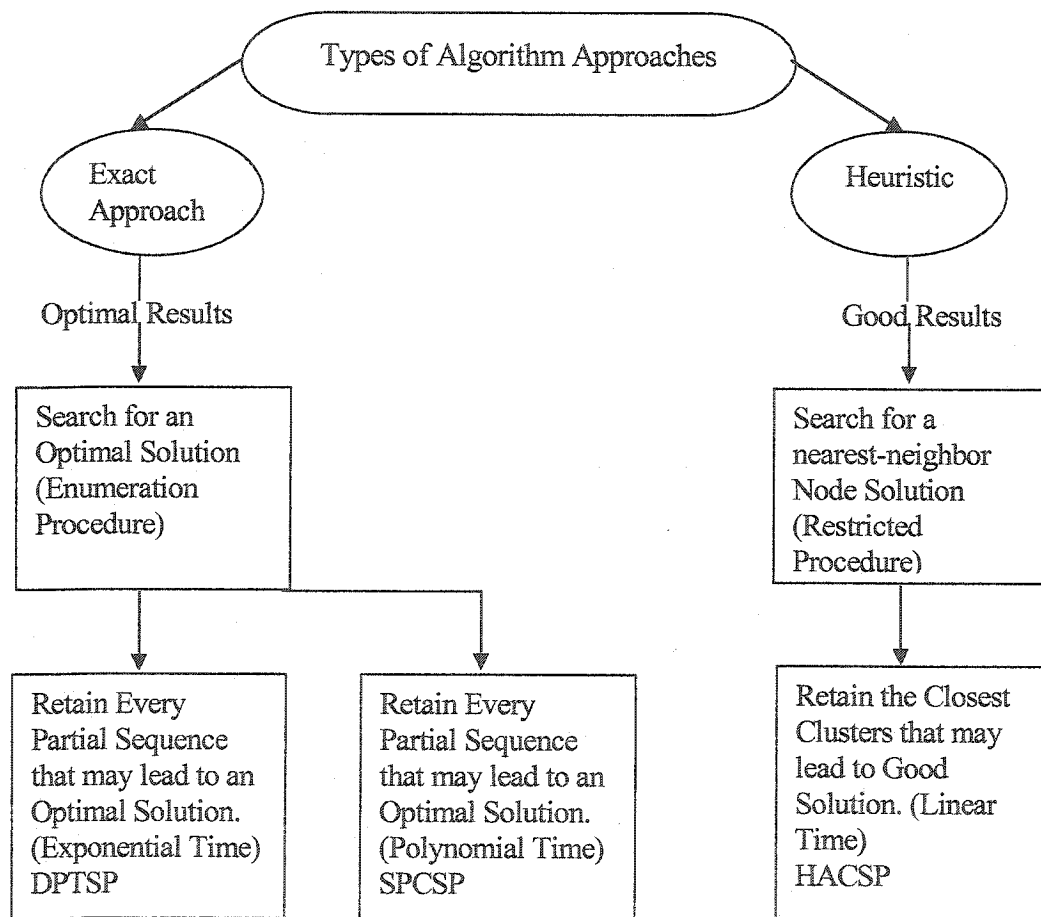
Problems with high computation costs need to accept feasible solutions, as opposed to optimal solutions. Therefore, it is feasible to develop a reasonable heuristic algorithm (approximations). A heuristic algorithm is a method used to solve a problem by trial and error; such a method can be valuable when it is expensive to use an exact approach, as in our case. Therefore, heuristic methods have a reasonable execution time and are helpful in increasing the convergence of exact algorithms. The last solution method used is the nearest-neighbor heuristic.

Finally, the developed simulation method is another approach that is needed to build the required stages and states for the problem. The simulation depends on the problem's number of clusters and can run in any optimization procedure. Its time complexity will follow the algorithm used to solve the problem. There will be a more detailed explanation about the developed simulation method and its characteristics to handle the cluster-sequencing problem in next sections.

**DPTSP:** Dynamic Programming for Travel Salesman Problem.

**SPCSP:** Shortest Path for Cluster Sequencing Problem.

**HACSP:** Heuristic Approach for Cluster Sequencing Problem.



**Figure 3:** The Proposed Solution Methods Approaches.



### 3.2 Mathematical Interpretation

For any transportation network, if there is no earthquake, the network has no damaged (closed) links or bridges and is therefore a baseline network configuration. But as a result of earthquakes, there are many damaged areas in the transportation network. Bridges and links are the most affected areas, and as a result, some will be closed. There are several clusters of damaged links. To simplify matters, we will make the assumption that there are three clusters: 1, 2, and 3. The immediate network configuration is  $\{1, 2, \text{ and } 3\}$ . We will also assume that repairs will start with Cluster 1. The configuration will be  $\{2, 3\}$  in this case. Conversely, assuming that Cluster 2 is repaired first, the configuration is  $\{1, 3\}$ . Finally, for repairs begun with Cluster 3, the configuration is  $\{1, 2\}$ .

Travel pattern (link volume and time) under network configuration  $\{2, 3\}$  should differ from that of configuration  $\{1, 3\}$ .

#### **Example:**

Let's use the three-cluster example. This example is applicable to all the solution methods used to solve the problem. Our objective is to reduce total system cost throughout the repair period to reach the baseline delay. We can define baseline

delay as the status of the transportation network where there is no damage to the system. In our example, there are six different sequences.

**Sequence 1: 1-2-3**

Baseline delay - {1, 2, 3}

Baseline delay - {2, 3}

Baseline delay - {3}

Baseline delay

**Sequence 2: 1, 3, 2**

Baseline delay - {1, 2, 3}

Baseline delay - {2, 3}

Baseline delay - {2}

Baseline delay

**Sequence 3: 2, 1, 3**

Baseline delay - {1, 2, 3}

Baseline delay - {1, 3}

Baseline delay - {3}

Baseline delay

**Sequence 4: 2, 3, 1**

Baseline delay - {1, 2, 3}

Baseline delay - {1, 3}

Baseline delay - {1}

Baseline delay

**Sequence 5: 3, 1, 2**

Baseline delay- {1, 2, 3}

Baseline delay- {1, 2}

Baseline delay- {2}

Baseline delay

**Sequence 6: 3, 2, 1**

Baseline delay - {1, 2, 3}

Baseline delay - {1, 2}

Baseline delay - {1}

Baseline delay

To calculate the objective value of each sequence, system cost must be calculated for the following 8 different network configurations:

Baseline delay - {1, 2, 3}

Baseline delay - {1, 2}

Baseline delay - {1, 3}

Baseline delay - {2, 3}

Baseline delay - {1}

Baseline delay - {2}

Baseline delay - {3}

Baseline delay

### 3.3 Method I: A Dynamic Programming Approach to the Modified TSP for the Cluster Sequencing Problem

To handle the research solution for the problem defined previously, there is a need for a method to reduce the number of network configurations resulting from the possible numbers of cluster combinations. This situation is suitable for dynamic programming. Therefore, to solve our problem, we will formulate an essential problem deriving from the vehicle routing problem, particularly with respect to the Travel Salesman Problem (TSP). The classical TSP can be used in our study with some important modifications. The TSP problem is defined as finding a tour in a complete graph of minimum cost. Furthermore, in the travel salesman problem definition, the salesman is required to visit each node of the transportation network once. In our case, the problem is asymmetrical and not symmetrical, meaning that:

$$i \rightarrow j \neq j \rightarrow i.$$

The cluster-sequencing problem is formulated as a dynamic programming approach to the modified TSP procedure due to many reasons. There are many identical or similar characteristics that the cluster-sequencing problem and the travel-salesman problem share. First, the cluster-sequencing problem consists of a number of clusters as nodes in the transportation network graph. As a result, to

solve the problem, there is a need to visit all the clusters once and only once. This will lead to attaining the required sequence for the specified clusters. It is obvious that this condition applies also to the standard travel-salesman problem. In addition, if the cluster-sequencing problem is solved as a modification from the travel salesman, we can find a benefit from the formulation. This approach is helpful to track a solution method for the cluster-sequencing problem. Furthermore, the graph shows cluster numbers as nodes, the solution method can give a specific weight for each cluster in the transportation network. Depending on the purpose, weights can vary in their meaning and values from one cluster to another.

By developing the algorithm via dynamic programming, large numbers of computations will be eliminated. This is because the solution algorithm for the dynamic programming will work in such a way that a minimized number of network analyses result from different network configurations. Also, dynamic programs are appropriate for use in the reconstruction and sequencing of repairs in transportation networks.

Furthermore, the solution is a dynamic optimization program because of the following two reasons:

- 1) The objective value should be the sum of each repair stage, output from each previous stage.
- 2) Each repair stage is affected by the previous sequence.

The assumption made in this step is that the traffic volume of a link is a function of the traffic volumes and status of other links in the network. By applying this concept, many researchers have developed traffic analysis models. The most popular model in the field is the user-equilibrium model. User equilibrium is used in the solution algorithm as a measure of transportation network performance. It can be assumed that the user equilibrium is equal to the distance variable in the vehicle-routing problem. It can also be assumed that, in addition to other factors, user equilibrium can be the scale of measurement in the cluster-sequencing problem.

Almost all methods available in the bridge-reconstruction field prioritize bridges at risk by summing weighted bridge structural and transportation characteristic scores. Also, the previous methods used for bridge sequencing did not utilize multistage formulation as a method, whereas our method scores the cluster, which includes bridges depending on importance and benefit to the transportation networks.

This solution method used to solve the cluster-sequencing problem differs from the classical TSP in several ways. First, it is a selective procedure. The transportation network graph for this algorithm shows that there are several links between each cluster and the next. Links in the figure represent the status of cluster repairs. Furthermore, links between any two clusters have different values of user equilibrium, as each link indicates a different number of clusters being

repaired. For instance, the links that connect Cluster A and Cluster B indicate the possible repair sequences that end by either Cluster A or Cluster B. So the algorithm procedure should differentiate between the different link values and choose the correct one.

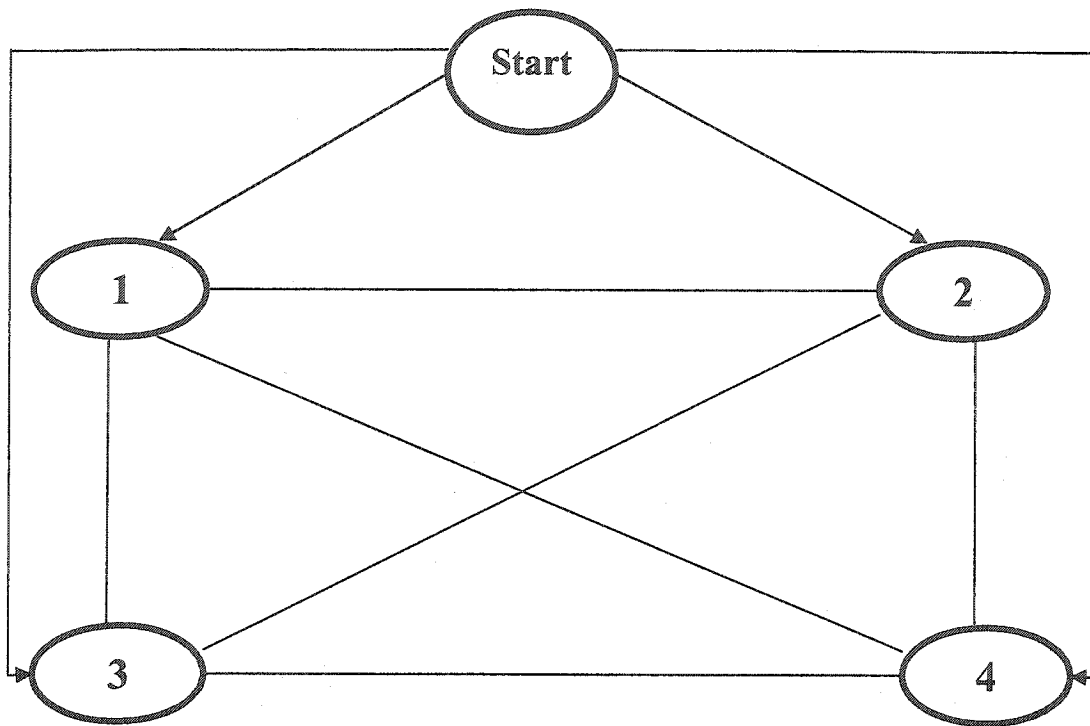
In addition, the algorithm applies the user equilibrium as an input for each link. User equilibrium is a result from the whole transportation network in each different configuration. Any user-equilibrium value depends on the sequence of cluster repairs.

Furthermore, the algorithm process should end when the last cluster is repaired. In this way, the algorithm will avoid forming a cycle, which in this case means to return to the “all clusters damaged” node.

Enumeration techniques such as dynamic programming reduce the number of nodes, i.e., the number of network configurations.

The problem in this figure is for 4 clusters with start node. It shows how many choices of the cluster repair sequences are available. Furthermore, the start node is the state in which all clusters are not repaired.





**Figure 4:** Four clusters plus start node are connected with explanatory links to show status of repairs.

### 3.3.1 Dynamic Programming Formulation

In dynamic programming, each stage consists of a finite number of states. If this is the case, the general formulation rule for the dynamic programming recursion (for a minimum problem) can follow this form:

$$f_t(i) = \min [(Result\ of\ current\ stage\ t) + f_{t+1}(\text{next state at stage } t+1)] \quad (1)$$

where  $f_t(i)$  is the minimum cost from stage  $t$  to the last stage  $T$ , and state  $i$  is state occurred at stage  $t$ .

There is a need to achieve the best recursion formulation of the dynamic program. For example, we should specify the relation of the objective function to our decision at a specific stage and its effect on that state as well as the subsequent state.

The following step is to define our stages, states, decision variable, and optimality recursions. We will then formulate the problem as a type of vehicle routing problem by applying dynamic programming to solve it.

The formulation introduced to model the cluster-sequencing problem (CSP) is an exact dynamic programming approach to the traveling salesman problem. The algorithm used for the CSP will demonstrate the benefits and

difficulties from the applied algorithm. The state variable in the cluster sequencing problem (CSP) depends on which cluster is repaired in the current stage and how many clusters were repaired in the last stages, or how many clusters still need to be repaired in the following stages.

The clustering sequencing problem (CSP) will be formulated by using the following equations. We will first introduce a general definition for the variable  $X_n$  such that:

$$X_n = \{ 0, 1 \} \quad (2)$$

Where  $X_1, X_2, \dots, X_k, \dots, X_n$  indicate the clusters indices that have been repaired up to the current stage. Therefore,  $X_n = 1$ , where cluster  $n$  is already repaired, and  $X_n = 0$  otherwise.

The cluster-sequencing problem is asymmetric and differs from the classical TSP in that the first repaired cluster in the tour will be repaired only once, whereas the TSP begins and ends with the same tour. As a result, we have added a constraint to avoid returning to the same cluster again.

$$X_{\text{first cluster in repair sequence}} = 1 \quad (3)$$

$f(Y; X_1, X_2, \dots, X_n)$  = measure of the minimum network delay from repairing cluster  $Y$  to last cluster  $X_n$  by repairing each cluster once and only once and repairing all available clusters.

According to Held and Karp (1962), Psaraftis (1978), and Bellman (1962) and his definition of “the principle of optimality,” which we have previously reported, we can formulate the cluster-sequence problem, with no preceding constraints, as follows:

$$f(Y_t; \{Z_t, X_t\}) = \text{Min}_{1 \leq K \leq n} \{UE + f(Y_{t+1}; \{Z_{t+1}, X_{t+1}\})\} \quad (4)$$

$$X_n = (X_1, X_2, X_3, \dots, X_k, \dots, X_n)$$

$n$  = Number of Clusters

$t$  = Stage Number = 0, 1, 2, ...,  $n$

$t+1$  = Next Stage Number, for  $\forall t \in \{0, 1, \dots, n-1\}$

$Y_t$  = The state variable that indicates the cluster number we are currently repairing at stage t.

$Y_{t+1}$  = The decision variable that indicates the cluster number we will repair at stage t+1.

Where stages are arranged as follows:

0, 1, 2, ..., n.

UE = Network delay or user-equilibrium from repairing cluster Y after a specified sets of repairs to clusters  $X_{1,2,...,k,...,n}$  and it means all travelers will choose routes that minimize their travel times and it will be achieved when travel time cannot be improved.

$Z$  = Vector including the clusters number that have been repaired,  
When  $X_Z = 0$

The following constraint is applied to ensure that the suitable link is used and as a subtour elimination constraint:

$$\sum_{n=0}^n X_t \geq \sum_{n=0}^n X_{t+1}, \quad (5)$$

$$\text{Where, } \sum_{n=0}^n X_t = \sum_{n=0}^n X_{t+1} + 1$$

When we solve the problem as a dynamic-programming formulation, we work backwards. And because the cluster-sequencing problem algorithm should end with the last repaired cluster, the algorithm will end when there is no cluster repaired. This constraint confirms that the algorithm will not have a cycle. This case is reached by the following:

$$\begin{aligned}
 f(Y_t; \{Z_t, [X_{nb}]\}) &= f(1; \{1, [0, 1, 1, 1, \dots, 1]\}) \\
 &= f(2; \{2, [1, 0, 1, 1, \dots, 1]\}) \\
 &\vdots \\
 &\vdots \\
 &\vdots \\
 &= f(n; \{n, [1, 1, 1, \dots, 0]\}) \tag{6}
 \end{aligned}$$

Furthermore, the baseline delay is achieved when we satisfy the following:

$$X_n = (1, 1, 1, \dots, 1) \tag{7}$$

As a result, this constraint means that all clusters in the transportation network are repaired and the algorithm will stop.

### **3.3.2 Number of possible cluster repair sequences formulation**

The problem consists of different stages and states. As a result, there are different links values between each pair of clusters. So depending on the repair sequence, links values will change to indicate different sequences.

Furthermore, we assume that each state in any stage has a unique UE value.

There is a need to develop a formulation for calculating the different number of links. So it is possible to estimate the number of different user equilibrium values.

As a result, values of user equilibrium indicate the different number of network configurations. Numbers of network configurations are the result of different cluster repair sequences. Clusters are formed as a way to repair groups of damaged bridges. Clusters can include any size of damaged bridges in transportation networks, depending on the characteristics specified to group the damaged bridges. As a result, different numbers of clusters can be formed for any transportation network.

As the number of different user equilibrium flows depends on how many cluster in the transportation network. Also, the number of links between any two clusters

in the figure depends on how many clusters exist in the transportation network.

Each link shows the repair sequences with the same user equilibrium value.

These repair sequences end by repairing any one of the two connected clusters.

The total number of links between any two clusters in the figure indicates the

number of different sequences for both connected clusters that come with

different user-equilibrium values. Furthermore, the purpose of these sequences is

to show how many connected cluster pairs can be arranged. In other words, each

link shows the number of sequences for these two connected clusters if one is

repaired after the other.

### **Example:**

For the case of 4 clusters there are 4 different values of user equilibrium between

any two clusters. Number of values is calculated as follow:

$$\binom{2}{0} + \binom{2}{1} + \binom{2}{2}$$

$$1 + 2 + 1 = 4 = C$$



$C$  = Indicates the number of possible cluster repair sequences (# of links) with different user equilibrium values between any 2 connected clusters. See Figures 5 and 6 in this case  $C = 4$ .

To compare the values of  $C$ , we will show another example that makes the concept clear. The example is for the case of 6 clusters. In this case, there are 16 different values of user equilibrium between any two connected clusters. Number of values follows this formulation:

$$\binom{4}{0} + \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4}$$

$$1 + 4 + 6 + 4 + 1 = 16$$

The general formula for the number of user equilibrium between any two clusters for any number of clusters in the transportation network will follow this mathematical formulation:

$$\sum_i^{n-2} \binom{n-2}{i} = \text{Different types of links between two clusters} \quad (8)$$

$\therefore$  Total number of links = [(Number of links between two clusters) \*  
(Number of connection links, edges)].

$$\text{Total number of links} = \left\{ \sum_{i=1}^{n-2} \binom{n-2}{i} * \left( \frac{n*(n-1)}{2} \right) \right\} \quad (9)$$

The different stages and each accompanying states for 4 clusters (example) are shown in Figure 6. Each differently colored link in the figure is considered to be a different stage and is also used to indicate the numbers of clusters that have been repaired. If we begin numbering the first stage as stage number (0), the last stage will be number (4). The number of nodes in the figure represents the number of clusters in the transportation network. The start node will represent the “all clusters damaged” transportation network delay.

The value on each link in Figure 6 is considered to be the user equilibrium value. It explains the sequence of repair for each cluster. It is necessary to calculate the user equilibrium after each cluster repair, for each different sequence of repair.

**Example:**

For any two clusters, the links' meaning is the same. Therefore, for the explanation issue, the 4 links between Clusters 1 and 2 are chosen. The explanation will be from up to down, as follows:

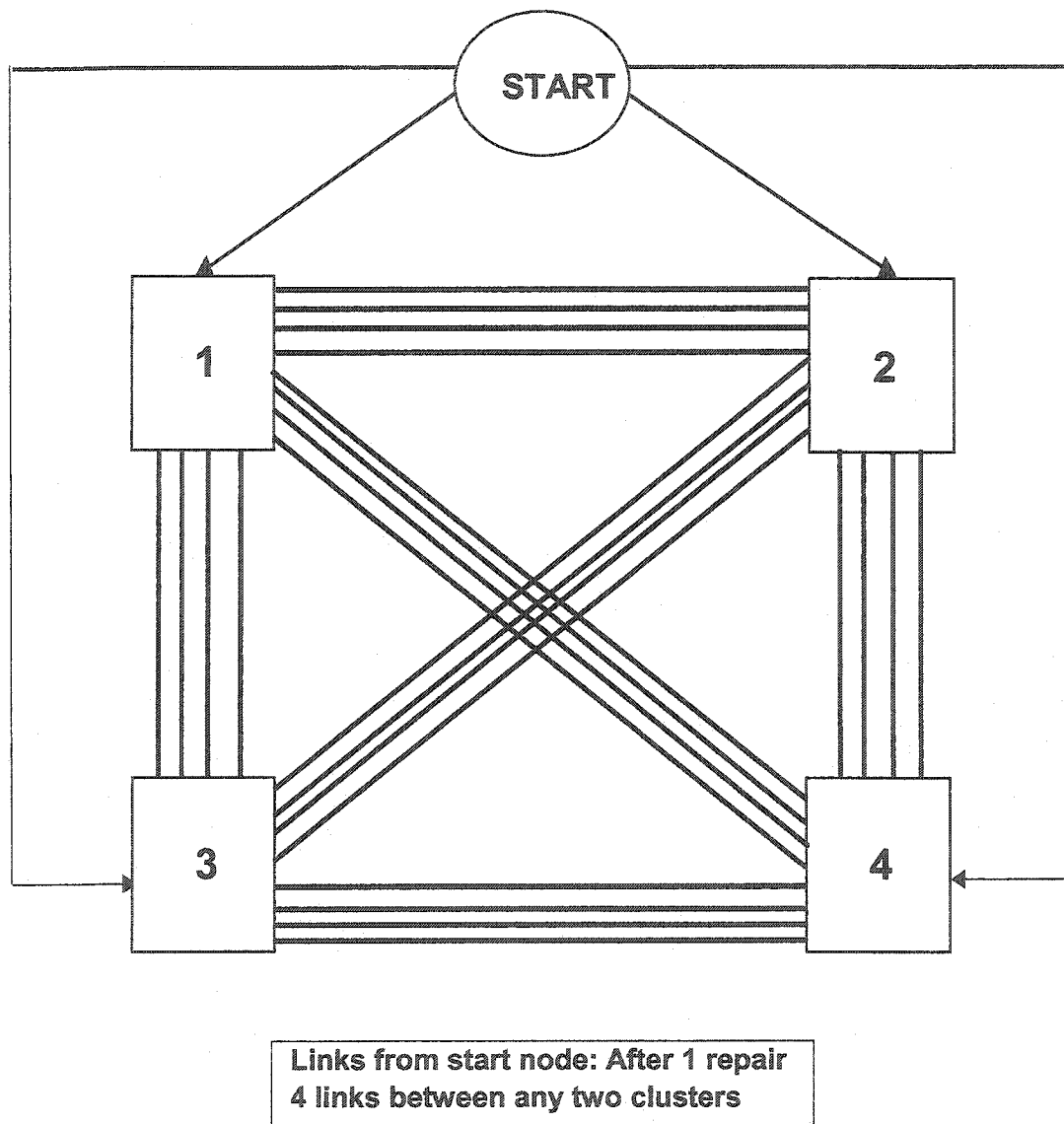
First Line: All possible sequences of  $X = (0,0,1,1)$  such that:  
Start-1-2 or Start-2-1.

Second Line: All possible sequences of  $X = (0,0,0,1)$  such that:  
Start-3-1-2 or Start-3-2-1.

Third Line: All possible sequences of  $X = (0,0,1,0)$  such that:  
Start-4-1-2 or Start-4-2-1.

Fourth Line: All possible sequences of  $X = (0,0,0,0)$  such that:  
Start-3-4-1-2 or Start-4-3-2-1 or Start-3-4-1-2 or  
Start-4-3-1-2.

**Figure 5:** The different choices of repair sequence for 4 clusters



**Figure 6:** The network used for different choices of repair sequence for 4 clusters

The difficulty of applying the dynamic programming approach to the modified TSP to the cluster-sequencing problem is due to that the problem belonging np-complete problems. The proposed formulation method gives an exponential time to solve the cluster-sequencing problem. Our goal is to track a solution to the cluster-sequencing problem by this formulation. Furthermore, it is assumed that our problem will handle a specific number of clusters. This assumption will not make the clusters size large.

More efficient calculations are required. As a result, there is a need to propose a competing algorithm procedure that can handle the cluster-sequencing problem in better than exponential time calculations.

### 3.4 Method II: Shortest Path Algorithm for the Cluster-Sequencing problem

In this section the shortest path algorithm is applied to the cluster-sequencing problem. One of the most reliable shortest path algorithms is Dijkstra algorithm. Dijkstra algorithm can handle the cluster-sequencing problem. Thus, it can be used to find the minimum total transportation network delay. It is a result of finding the minimum transportation network delay between any two clusters. The shortest path method has huge implications for the fields of operations research and combinatorial optimization, as many realistic problems belong to the same structure that the shortest path uses. Furthermore, the shortest path problem can be formulated in a step-by-step method. In this way, the concept for any optimization problem solved by the shortest path algorithm can be clear and explainable. Also, as mentioned in earlier sections, if the cluster-sequencing problem is considered to be a type of combinatorial optimization problem, then it can be formulated as a shortest path problem.

Dijkstra algorithm is one of the best known and most implemented of the shortest path algorithms. The reason is that it is efficient and simple to use when handling many optimization problems. The idea for Dijkstra's algorithm is that it can follow straightforward procedures. Let's assume that the optimal repair sequence for the clusters as follows: "All clusters not repaired state,..., Cluster: k,..., Cluster: n". If this repair sequence represents the shortest path, then each

segment length between any two clusters results from the better comparison for those two clusters.

The following sections will explain how the cluster-sequencing problem can be solved as a shortest path problem—specifically, how Dijkstra algorithm can handle the cluster-sequencing problem.

### **3.4.1 Cluster-sequencing problem as a shortest path problem**

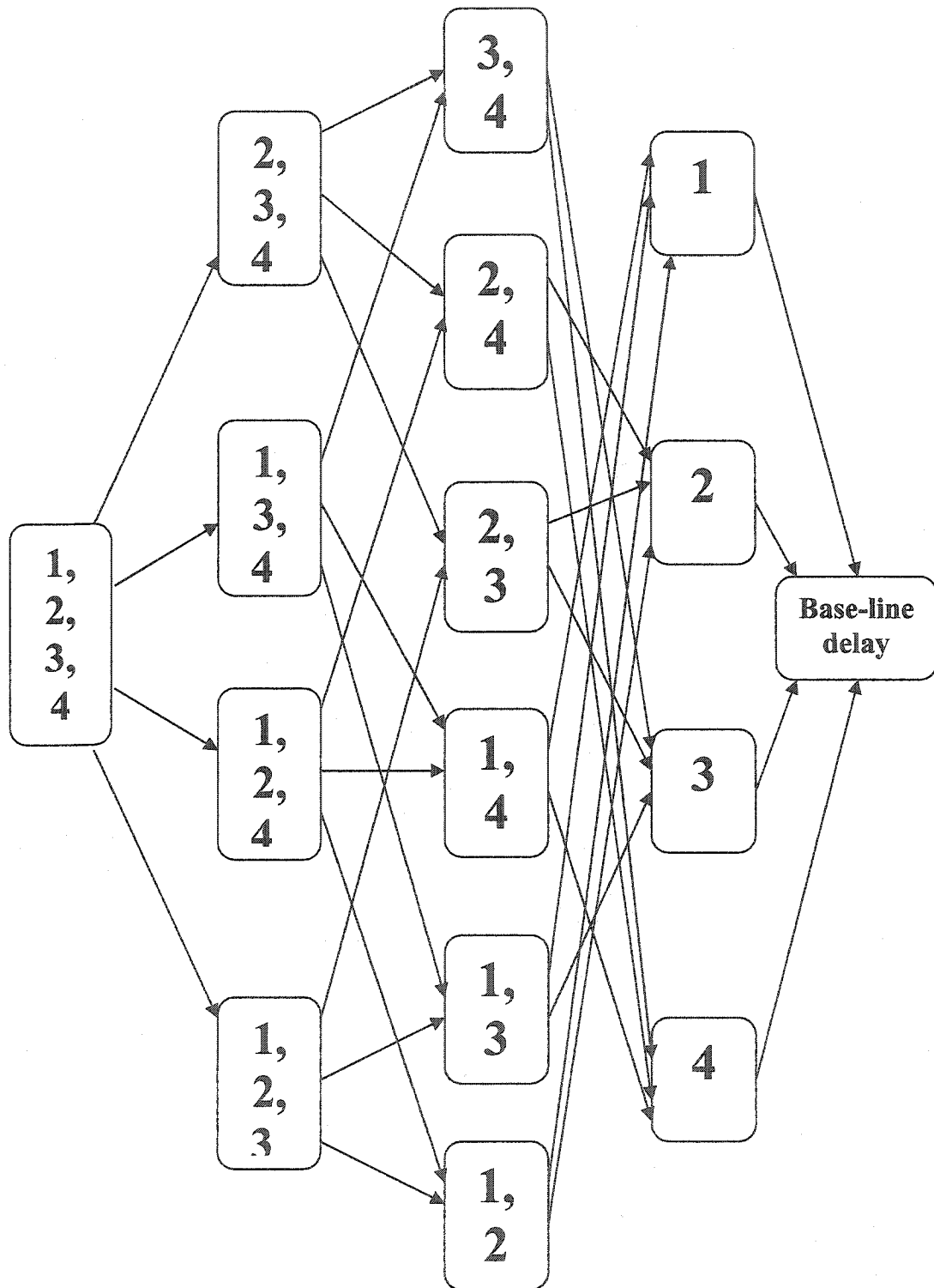
The cluster-sequencing problem can be shown in a graph as a transportation network. Figure 7 shows how the cluster-sequencing problem can follow the classical graphs of shortest path algorithms. The figure defines the different repair sequence for an example of 4 clusters. The different stages and all accompanying states for 4 clusters (example) are shown in Figure 7. Each column of blocks in the figure is considered to be a different stage. The stage is defined as one cluster repair is added to the existing cluster repairs. The final stage is achieved when the last cluster is repaired. If we begin numbering the first stage as stage number (0), the last stage will be number (4). The numbers of blocks in each stage are the states of the transportation network. The state for the network is defined by the number of clusters that have been repaired or the number of

clusters still to be repaired. Furthermore, the state for a cluster is defined as whether the cluster requires repair, or whether its repair will maximize the return.

The value on each link in Figure 7 is considered to be the user equilibrium value. It explains the sequence of repair for each cluster. The user equilibrium values that belong to the shortest path will give the total minimum delay in the transportation network.

For any number of clusters to be handled as a shortest path algorithm, it is required to show all the varying cluster repair states. Figure 7 shows the design of shortest path algorithms.





**Figure 7:** Number of states for 4 Clusters

The general formulation for any number of clusters to be transferred to a classical shortest path algorithm is as follows:

Number of stages =  $n + 1$ ,

where  $n$  = Number of clusters and stage numbers from stage 0 to stage  $n$ .

$$\text{Number of states in stage } t = \binom{n}{t} = \left( \frac{(n)!}{(n-t)! * (t)!} \right) \quad (10)$$

Total Number of States = Number of user equilibrium to be calculated

$$= \sum_{t=0}^{t=n} \binom{n}{n-t} = 2^n \quad (11)$$

By using these equations, it is possible to show the transportation network for any given number of clusters.

### **Example:**

If there are 4 clusters to repair in the transportation network, as in Figure 7, then numbers of stages are as follow:

$$(n - 4), (n - 3), (n - 2), (n - 1), (n)$$

$$\text{Number of states in stage } (n - 4) = \binom{4}{0} = 1$$

$$\text{Number of states in stage } (n - 3) = \binom{4}{1} = \left( \frac{4!}{3! * 1!} \right) = 4$$

$$\text{Number of states in stage } (n - 2) = \binom{4}{2} = \left( \frac{4!}{2! * 2!} \right) = 6$$

$$\text{Number of states in stage } (n - 1) = \binom{4}{3} = \left( \frac{4!}{1! * 3!} \right) = 4$$

$$\text{Number of states in stage } (n) = \binom{4}{4} = 1$$

### 3.4.2 Pseudo-code for Shortest Path Algorithm Used for the Cluster-Sequencing Problem

Dijkstra algorithm—which can be considered a “greedy” algorithm—tries to find a local optimal solution. The algorithm chooses the closest cluster and adds the result to the last calculated solution. Hence, after using the same process several times, the final solution will give a global optimal. In general, this method will not give a global optimal for all greedy algorithms, as will be shown in next section. But it is applicable to Dijkstra algorithm, and the global optimal is achieved.

Cluster-sequencing problem using Dijkstra Algorithm “pseudo-code”:

The mathematical formulation for the cluster-repair problem follows these code steps:

Initiate cluster-sequencing problem network

```
{
// Initialize costs
Label all cluster as not repaired
For clusters from 0 to n-1
{
    Assign a set S for the NR state
    Another set V = all states except NR state
    Use the parent node or the origin
    D [i] = Cost [NR, i]
}
For clusters from 0 to n-2
{
// Repeated n-1 times
    Initialize Dk = INFINITY
    For j = 0 to n-1
// To pick the min k in V without S
```

```

Select min D [j]
Dk = D [j]
// Update the costs
Label repaired cluster, k
For v = 0 to n-1
D [v] = min (D [v], D [k] + Cost [k, v])
Give cluster repair sequence
}
}

```

NR = Nothing repaired state

V = Clusters in the transportation network

S = Clusters that are visited

n = Number of clusters available in the study area

D = Result after each different repair sequence

Cost = The connection between two stages in two different states

The coded algorithm returns the total network performance that is measured by using the user equilibrium. In addition, the code returns the cluster repair sequence.

A more efficient running time is the result of applying the Dijkstra algorithm to the cluster-sequencing problem. The running time is much less than it was with the modified TSP used in the last sections.

### 3.4.3 The Total Delay in the Transportation Network

It is important for decision makers to estimate the reduction in delay resulting from the repair of any cluster in the transportation network. The reduction in delay from repairing a given cluster is defined as the time of traveling between any origin and destination across the transportation network. It can be calculated as a monetized benefit from each cluster.

For example, let cluster number 1 be the last cluster in the repair sequence. The baseline state will be reached when the last cluster is repaired. The reduction in travel time that results from repairing this last cluster is calculated as follows:

$$\begin{aligned} \text{Savings from cluster \# 1 repair} &= \$ \text{ of repairing cluster \# 1} \\ &\quad - (\text{Delay \# 1} - \text{Baseline}) \end{aligned}$$

$$\text{Delay \# 1} > \text{Delay baseline}$$

The calculation of savings from any other cluster repair will follow the same procedure.

$$\begin{aligned} \text{Savings form cluster \# n repair} &= \$ \text{ of repairing cluster \# n} \\ &\quad - (\text{Delay \# n} - \text{Baseline}) \end{aligned} \quad (12)$$

$$\text{Delay \# n} > \text{Delay baseline} \quad (13)$$

The important of all calculations is to reach the case where no clusters are to be repaired. This case is called the baseline. So it is required to know the baseline delay.

$$\begin{aligned} \text{Total Network Delay (complete network)} &= \sum_a t_a (V_a)^* V_a \\ &= \text{Baseline delay} \end{aligned} \quad (14)$$

$V_a$  = Volume of the transportation network

$t_a$  = Travel time

Baseline delay = Total system cost

### **3.5 Method III: The Heuristic Approach for the Cluster-Sequencing Problem**

The cluster-sequencing problem was formulated as an exact approach in all last optimization algorithms. Exact algorithms give very good results with high consumption of computation time. The modified TSP algorithm used in our last sections carries an exponential running time. This computation difficulty constraints the size of the problem and the computing resources, unless it is available the sufficient resources to handle each cluster repairs simultaneously.

Our goal is to examine the most suitable algorithms to solve the cluster-sequencing problem, as is mentioned in earlier sections. In the last sections, we modeled the optimization algorithms that vary in the computation time and give a global result. It is highly advantageous to attain the global results for the CSP from these optimization algorithms. But sometimes decision makers, planners, and the concerned authorities require preliminary or estimated results. The need for preliminary studies highlights the importance of these preliminary results. It is an important issue to have a general and fast estimate for the repair costs of clusters, the transportation network performance, and the other related emergency issues. Results such as these can be implemented for an emergency plan for an evacuation of the affected area.

So there is a need for a method that can deal with our problem in the cheapest expenses applications. One of the available and recommended methods for the



purposes mentioned earlier is the greedy method. This algorithm follows the cheapest cluster repair. It is suggested that this greedy algorithm build the cluster repair sequences by comparing the results of the transportation network delay. The network delay comes from different network configurations. The algorithm will repair the least expensive cluster in each stage. This means the most saving in network delay is achieved in each specific stage of repairs. So the cluster that improves the network performance is chosen.

In the heuristic approach, there is a saving in the amount of computations but a decreasing in the quality of the results. A greedy algorithm sometimes works well for optimization problems. The Dijkstra algorithm solves the cluster-sequencing problem as a greedy search but builds the calculations to end with a global result. The heuristic approach—from the beginning state where no cluster is repaired, to the ending state where all clusters are repaired—is summarized as follows:

- Start from initial state  $(1, 1, 1, \dots)$
- Leave initial state to one of these states:  
 $(0, 1, 1, \dots), (1, 0, 1, \dots), (1, 1, 0, \dots), \dots, (1, 1, 1, \dots, 0)$  depending on cheapest/shortest result.
- The cheapest result is chosen depending only on the current stage.

- Leave next to a new cluster repair state after one cluster has been repaired using the same condition constraint by which last state is reached.
- Continue until the last state (0, 0, 0, ...) is reached.

### 3.5.1 Cluster-sequencing problem using the heuristic approach the

#### “Pseudo-code”

The heuristic approach is formulated as a step-by-step code to deal with cluster repair. It follows these formulation steps:

Initiate cluster-sequencing problem graph

```
{
// Initialize costs
Label all clusters as not repaired

For stages from 0 to n-1
{
    Initialize cost to be = INFINITY

    For j = 0 to n-1
    {
```

Choose the cheapest link from the current state to a state in the next stage between each two stages

Set the cost of repairs = Cost of all repaired clusters + current cluster in repair.

Update visited clusters

Add cost to visited cluster repair sequence

}

Give cluster repair sequence

}

$n$  = Number of clusters available in the study area

Cost = the connection between two stages in two different states

### 3.6 Binary Numbers representing status for cluster repair

The different states in the transportation network, which represent the different cluster sequences, are explained by using a binary numbers concept. The idea is to show a different 0 or 1 vector representing each different cluster repair sequence. In the code matrix, the binary arrangements are used to clarify the meaning of which cluster is repaired. This method is needed to build each stage and its accompanying states graph. The binary numbers procedure accommodates any number of clusters. (1 = Repaired, 0 = Not Repaired).

For example, when there are 3 clusters to be repaired, the vector (1, 0, 1) = cluster number 2 not repaired and clusters 1 and 3 are repaired.

Figure 8 shows the concept of the binary numbers and how it represents the entire cluster repair sequence.

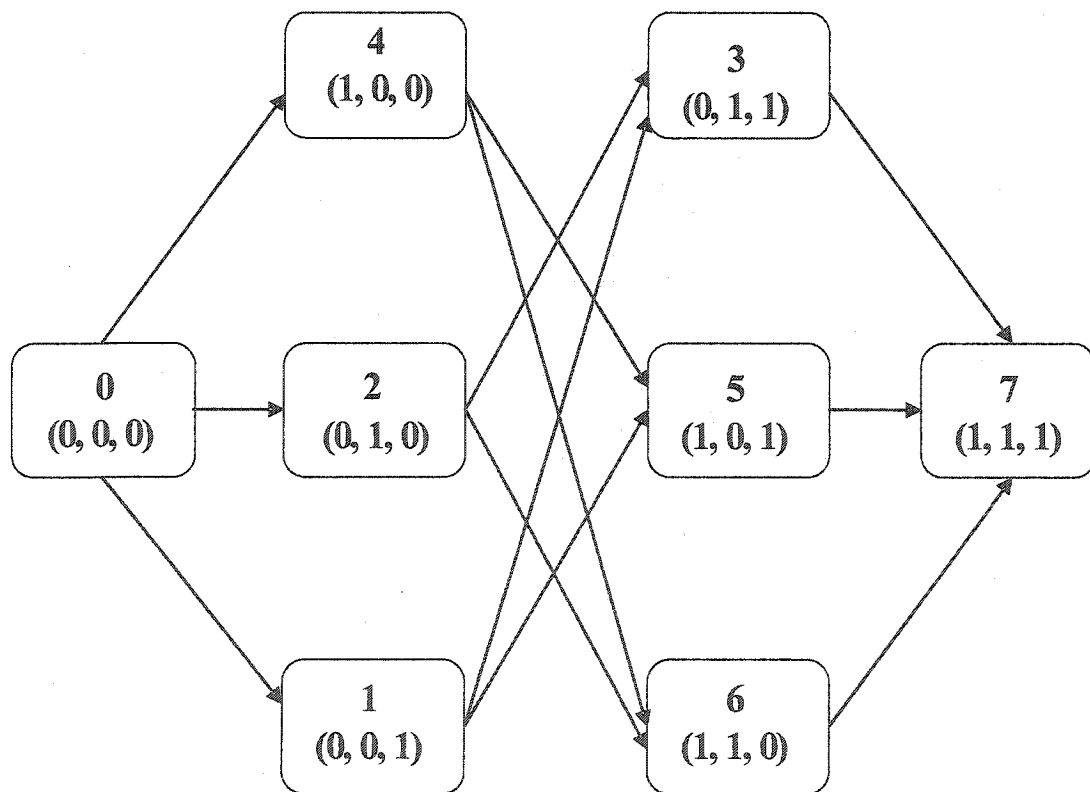
There are many variables introduced in the formulation of the binary method. All of them are shown in Appendix D. One of the important variables is the weight variable. It is used to find the weight of node  $i$  = number of 1's in  $i$  as a binary.

$$wt(x) = \begin{cases} 0 & , X = 0 \\ X \bmod 2 + wt\left(\frac{x}{2}\right) & \end{cases} \quad (15)$$

**Example:**

- State 26 = (1, 1, 0, 1, 0) = ( $2^4 + 2^3 + 2^1$ )
- i.e. State 26 has clusters 1,2 and 4 repaired

Figure 8 shows an example of how can it look like the network for 3 clusters by using the binary numbers method.



**Figure 8:** Binary Numbers represent the different cluster repair sequences

#### **4. Examples and Analysis of Cluster-Sequencing Problem Solution Methods**

This chapter includes the results of some applied algorithms to the cluster-sequencing problem. An explanation of the results achieved from both the Dijkstra algorithm and the heuristic approach are discussed. The formulation of the dynamic programming approach to the modified TSP algorithm carries an exponential running time. The shortest path algorithm used to solve the cluster-sequencing problem leads to exact results that give optimal answers. The heuristic approach cannot guarantee globally optimal results. A good, reasonable solution from a heuristic approach the optimal is needed sometimes even in the problem cannot be solved optimally. This method handles problems of all sizes with less computation time than all the last solution methods.

Examples of different characteristics are shown for the purpose of comparison and analysis between the solution methods. In general, the results from the applied algorithms will be different.

## **4.1 Applying the cluster-sequencing algorithms in a specific example**

The following examples show the repair sequence for different algorithms used in the cluster-sequencing problem. The purpose is to prove that the cluster-sequencing problem is solvable and there is a solution from applying these algorithms.

### **4.1.1 Examples for the shortest path and heuristic approach algorithms**

The purpose of the following theoretical examples is to test the solution from the algorithms used to solve the cluster-sequencing problem. The examples show the clusters repair sequence and total travel time for the transportation network.

In the first example, there are 3 clusters to be examined. Then there is another example that will apply the algorithms for 4 clusters. The purpose in all the examples used is to see the difference between each algorithm results. Each algorithm will be applied to the cluster-sequencing problem and give the cluster repair sequence. There are differences in the cluster repair sequence results that depend on the type of the algorithm. Therefore, there is a need to have a section that compares between both algorithms results. This section will follow these examples and will include all the details.

**Example:**

Number of clusters: 3

Size of the graph: 8 states

The values in each state are as follows:

<b>Stage 1</b>		
<b>State # 1</b>	<b>State # 2</b>	<b>State # 3</b>
<b>50</b>	<b>68</b>	<b>77</b>
<b>Stage 2</b>		
<b>State # 4</b>	<b>State # 5</b>	<b>State # 6</b>
<b>44</b>	<b>35</b>	<b>42</b>
<b>Stage 3</b>		
<b>State # 7</b>		
<b>15</b>		

**Table 2:** Values of the states for 3 clusters



**Shortest path Results:**

**The sequence is:**

**State # 0 = (0, 0, 0),**

**State # 1 = (1, 0, 0),**

**State # 5 = (1, 1, 0),**

**State # 7 = (1, 1, 1).**

**Total cost = 100**

**Heuristic Approach Results:**

**The sequence is:**

**State # 0 = (0, 0, 0),**

**State # 1 = (1, 0, 0),**

**State # 5 = (1, 1, 0),**

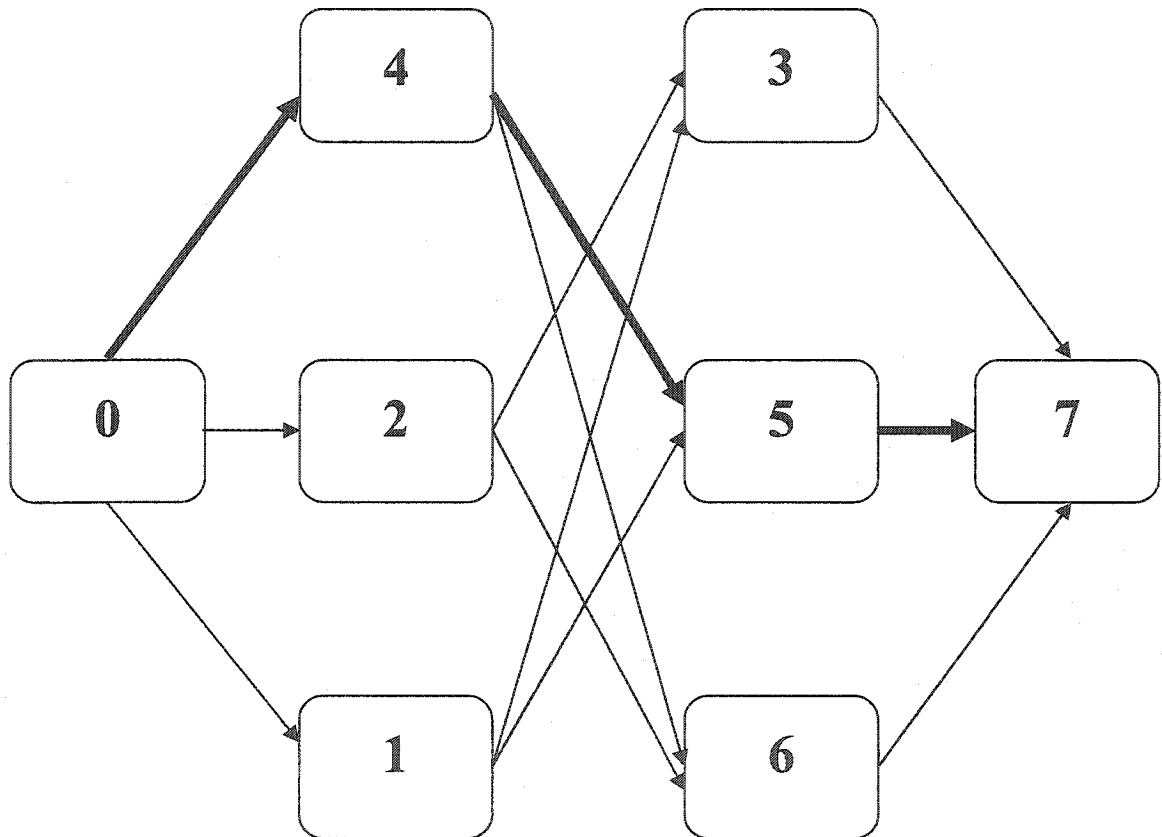
**State # 7 = (1, 1, 1).**

**Total cost = 100**

<b>Stage 1</b>		
<b>State # 1</b>	<b>State # 2</b>	<b>State # 3</b>
(1, 0, 0)	(0, 1, 0)	(0, 0, 1)
<b>Stage 2</b>		
<b>State # 4</b>	<b>State # 5</b>	<b>State # 6</b>
(1, 1, 0)	(1, 0, 1)	(0, 1, 1)
<b>Stage 3</b>		
<b>State # 7</b>		
(1, 1, 1)		

**Table 3:** States represented in binary numbers for 3 clusters to indicate cluster conditions.

The last example shows how both algorithms applied to the cluster-sequencing problem give similar results. It is possible sometimes for the heuristic approach algorithm applied to handle small-size clusters with excellent results. The heuristic approach is smart enough to track the best paths for non-complex problems.



**Figure 9:** States as numbers for 3 clusters and results from the algorithms

**Example:**

Number of clusters: 4

Size of the graph: 16 states

The values in each state are as follows:

Stage 1					
State # 1	State # 2	State # 3	State # 4		
50	60	77	65		
Stage 2					
State # 5	State # 6	State # 7	State # 8	State # 9	State # 10
44	48	42	41	47	46
Stage 3					
State # 11	State # 12	State # 13	State # 14		
34	36	35	29		
Stage 4					
State # 15					
22					

**Table 4:** Values of the states for 4 clusters

**Shortest path Results:**

**The sequence is:**

**State # 0 = (0, 0, 0, 0),**

**State # 1 = (1, 0, 0, 0),**

**State # 5 = (1, 1, 0, 0),**

**State # 14 = (1, 1, 1, 0),**

**State # 15 = (1, 1, 1, 1).**

**Total cost = 145**

**Heuristic Approach Results:**

**The sequence is:**

**State # 0 = (0, 0, 0, 0),**

**State # 1 = (1, 0, 0, 0),**

**State # 7 = (1, 0, 0, 1),**

**State # 13 = (1, 1, 0, 1),**

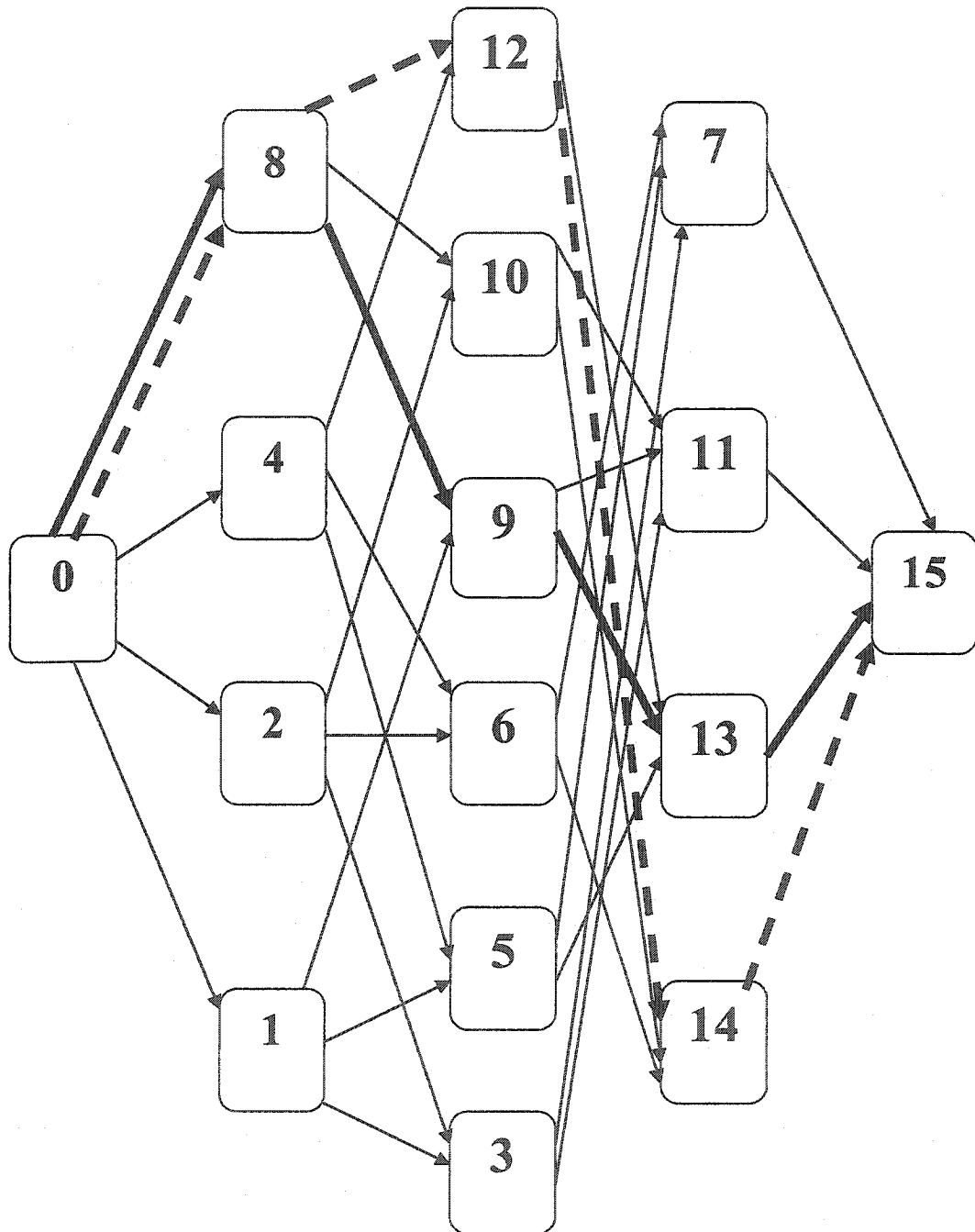
**State # 15 = (1, 1, 1, 1).**

**Total cost = 149**

The following table shows the different cluster states. Its values vary depending on each stage and state. It is assumed that there is only one cluster repair in each stage.

Stage 1					
State # 1 (1, 0, 0, 0)	State # 2 (0, 1, 0, 0)	State # 3 (0, 0, 1, 0)	State # 4 (0, 0, 0, 1)		
Stage 2					
State # 5 (1, 1, 0, 0)	State # 6 (1, 0, 1, 0)	State # 7 (1, 0, 0, 1)	State # 8 (0, 1, 1, 0)	State # 9 (0, 1, 0, 1)	State # 10 (0, 0, 1, 1)
Stage 3					
State # 11 (0, 1, 1, 1)	State # 12 (1, 0, 1, 1)	State # 13 (1, 1, 0, 1)	State # 14 (1, 1, 1, 0)		
Stage 4					
State # 15 (1, 1, 1, 1)					

**Table 5:** States represented in binary numbers for 4 clusters to indicate cluster conditions.



**Figure 10:** Shortest path method result in dash arrows, and heuristic approach algorithm result in bold arrows.

#### **4.2 Results of the traffic assignment from the study area**

The study area is the Bay Area. The traffic assignment is applied to the specific area in the Bay Area. The TransCAD software gives all the needed results. See Appendix B. It is tested for 3 clusters of damaged bridges. The results give a general idea about the transportation network conditions after each different network configuration. The TransCAD software results are used to define the inputs of the algorithms of the cluster-sequencing problem.

Depending on the size of the clusters given in the beginning of the problem, each transportation network will vary. The example handled in the TransCAD software consists of 3 clusters. Each cluster contains a different numbers of damaged bridges and, hence, a different number of closed links. In each network configuration, there are different links and bridges that are closed or open to traffic. In the appendix, the table shows the results where all clusters are damaged (nothing repaired) and the case where all clusters are repaired.



### 4.3 Testing Network Configurations Results by a Simulating method

Damaged-bridge clusters are an essential method to optimize the transportation network performance, as mentioned in earlier sections. Clusters are then needed to be repair in the most efficient way. This is due to the many major traffic problems that will exist after an earthquake. The main objective is to see what the best algorithm for the problem is and to compare between the applied algorithms. Therefore, there is a need for a method that applies both algorithms in several transportation network conditions. Furthermore, there is a need to know the difference between the results for the applied algorithms.

The developed simulation method represents the travel time results for each transportation network configuration. The output of the simulation method makes some assumptions. These assumptions are necessary to make the model similar to the real results case. As the results are calculated from the TransCAD software for the Bay Area, it is good to build the simulation model depending on the real one. Therefore, the assumptions applied to the simulation model are non trivial. It is assumed that after each cluster repair, the network delay will improve. So the simulation method will generate results depending on how many clusters are repaired. In other words, the simulation results depend on the stage number. The other assumption is that the final result achieved from repairing all the clusters is the baseline network delay.

A huge savings of the calculation of each transportation network configuration is achieved through the network's simulation results. It is due to that all the different network configurations are simulated. In this procedure, more analysis techniques can be applied. This is due to the availability of the different results that represent different network configurations.

The main concept that the simulation method depends on is the concept of binary numbers. The binary numbers are used in a way that converts each different network configuration to be represented only in 0 and 1. All the states in the transportation network carry different node number. Each of these node numbers is solely explained in 0 and 1. For example, in the case of 3 clusters of damaged bridges, the node number 4 or state number 4 is equal in the binary numbers to (1, 0, 0).

There are many variables introduced in the formulation of the simulation method. All of them are shown in Appendix D. One of the important variables is the weight variable. It was explained in the binary section. It is used to find the weight of node  $i$  = number of 1's in  $i$  as a binary.

$$wt(x) = \begin{cases} 0, & X = 0 \\ X \bmod 2 + wt\left(\frac{x}{2}\right) \end{cases}$$

### 4.3.1 Examples from the simulation method

#### Example:

This example shows the results of the repair sequence for 3 clusters. It is shown that both shortest and heuristic algorithms are giving the same repair sequence. As is mentioned earlier, it is possible that both algorithms give the same results. Depending on the structure of the simulation, it is assumed that the heuristic algorithm is smart enough to track the results that are simulated.

3 Clusters: <i>Heuristic</i> <i>Results:</i>				<i>Shortest</i> <i>Results:</i>			
Run #	Sequence by Total of Binary #	Binary number Representation	Sequence by Cluster #	Run #	Sequence by Total of Binary #	Binary number Representation	Sequence by Cluster #
1	0	(0, 0, 0)	0	1	0	(0, 0, 0)	0
	2	(0, 1, 0)	2		2	(0, 1, 0)	2
	6	(1, 1, 0)	3		6	(1, 1, 0)	3
	7	(1, 1, 1)	1		7	(1, 1, 1)	1

**Table 6:** Simulation results for 3 clusters.

### Example:

This example will show the difference between the applied algorithms results. It is applied to 5 clusters.

5 Clusters: <i>Heuristic</i> <i>Results:</i>				<i>Shortest</i> <i>Results:</i>			
Run # 1	Sequence by	Binary number	Sequence by	Sequence by	Binary number	Sequence by	
	Total of Binary #	Representation	Cluster #	Total of Binary #	Representation	Cluster #	
	0	(0, 0, 0, 0, 0)	0	0	(0, 0, 0, 0, 0)	0	
	16	(1, 0, 0, 0, 0)	5	16	(1, 0, 0, 0, 0)	5	
	24	(1, 1, 0, 0, 0)	4	24	(1, 1, 0, 0, 0)	4	
	26	(1, 1, 0, 1, 0)	2	28	(1, 1, 1, 0, 0)	3	
	27	(1, 1, 0, 1, 1)	1	29	(1, 1, 1, 0, 1)	1	
	31	(1, 1, 1, 1, 1)	3	31	(1, 1, 1, 1, 1)	2	
Total = 767				Total = 759			

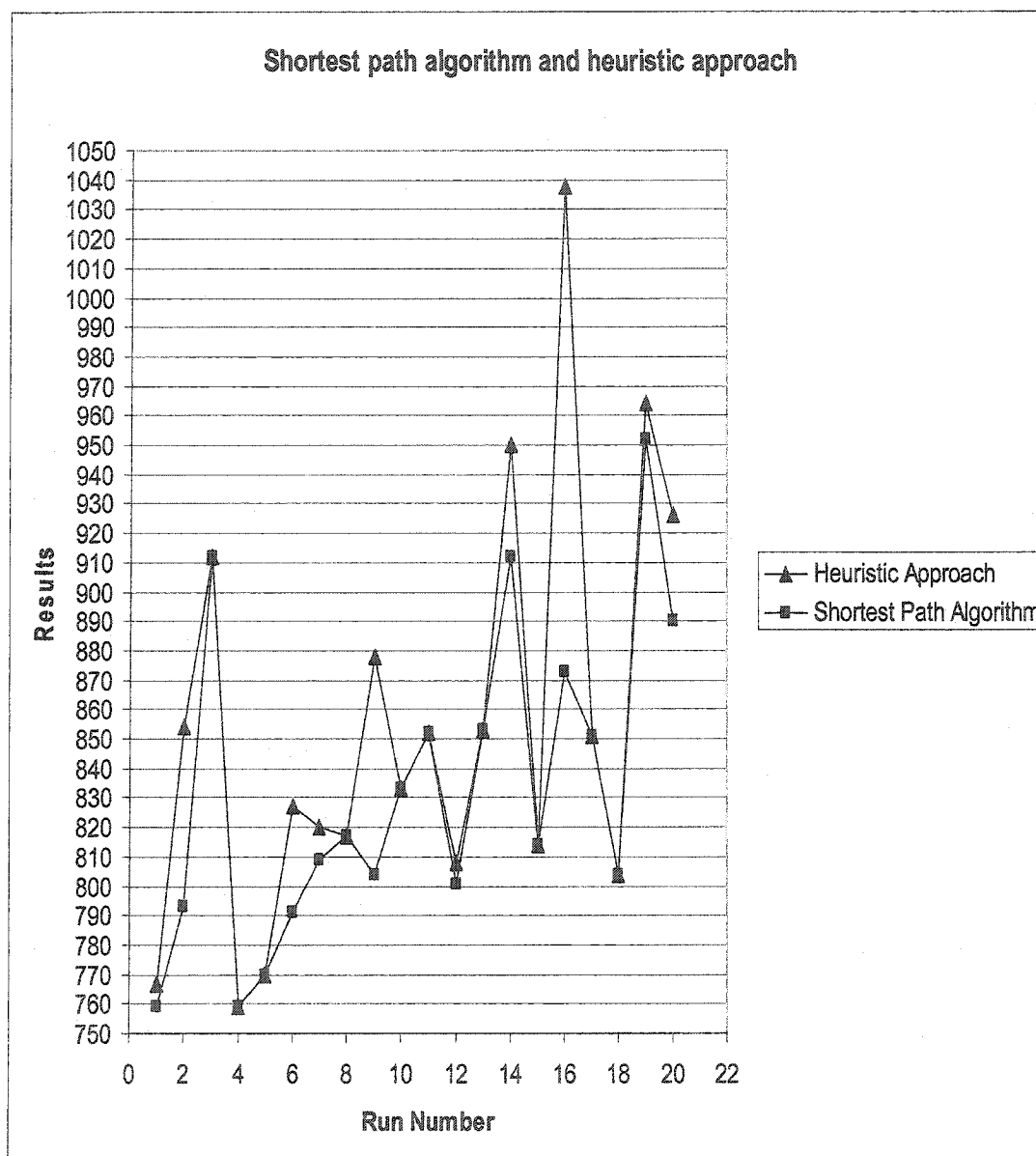
**Table 7:** Different results for the simulating method in 5 clusters

**Example:**

The several simulation runs will generate different results in most cases of applying the algorithms. It is due to different clustering design. Then it is obvious that each different resulting number means a different repair sequence.

<b>5 Clusters:</b>	<b>Heuristic Results:</b>	<b>Shortest Results:</b>
<b>Run #</b>		
1	767	759
2	854	793
3	912	912
4	759	759
5	770	770
6	827	791
7	820	809
8	817	817
9	878	804
10	833	833
11	852	852
12	808	801
13	853	853
14	950	912
15	814	814
16	1038	873
17	851	851
18	804	804
19	964	952
20	926	890

**Table 8:** The results of each algorithm for different runs



**Figure 11:** The differences between the applied algorithms

#### 4.3.2 Testing different number of clusters

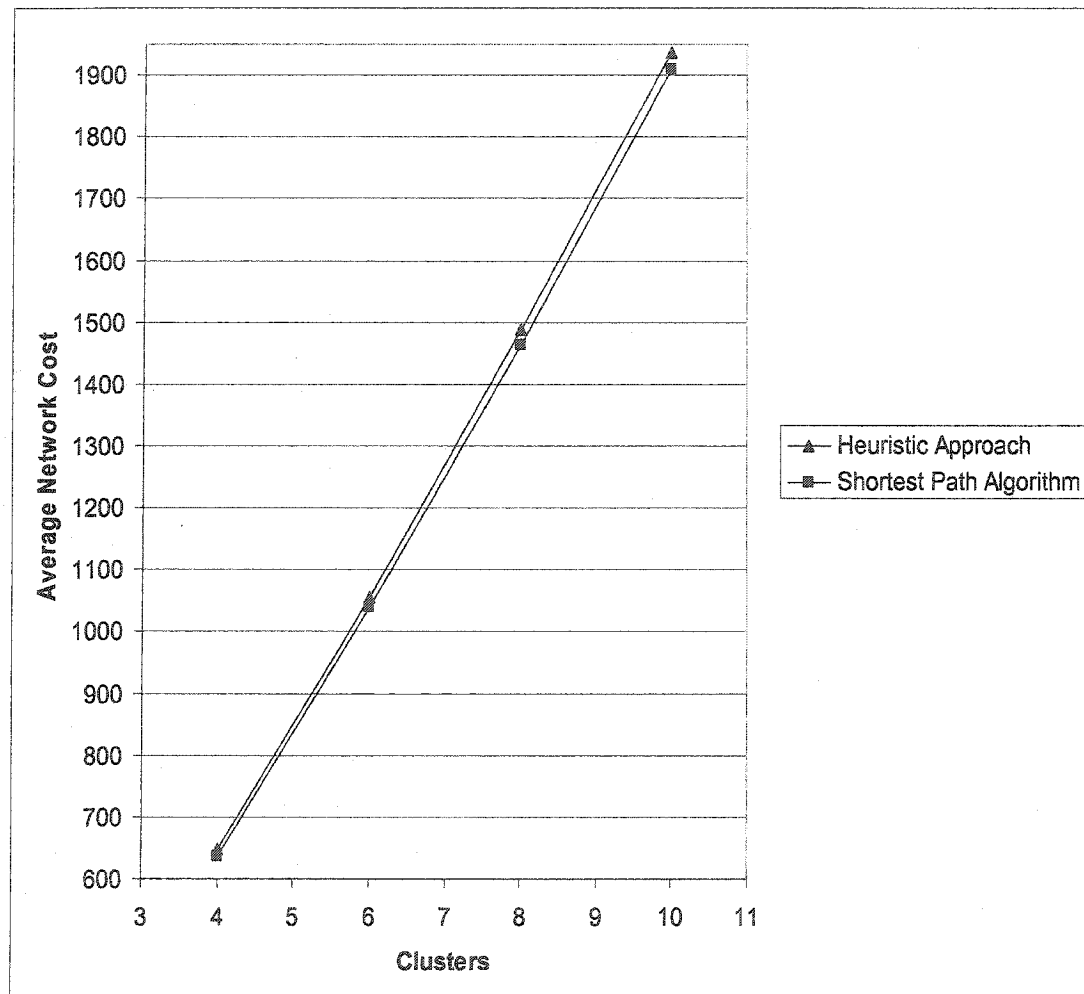
Testing a wide range of results will return a big picture about the differences between the tested algorithms. Therefore, it is good to see the different curves for the tested algorithms. For that purpose, the program is tested for several runs. Each specific range of runs is a result of definite number of clusters. Then the results are implemented in a graph to show the shape of each algorithm curve. These types of tests are helpful for the decisions purposes regarding the accuracy of the algorithms applied. It shows which algorithm, for each number of clusters, is the most feasible to solve the problem. This study gives a future plan for the decision makers. They will have an estimate of the differences between these algorithms for each different number of clusters.

Figure 12 shows that results from algorithms applied for the cluster sequencing problem are similar for a small number of clusters. But when the number of clusters is big, the difference in results between both algorithms is increased. In general, the overhead is almost the same for all the different number of clusters tested. It is between the ranges of change of %1.3 to %1.7 only.

# of Clusters	Average of Heuristic Approach Results:	Average of Shortest Path Algorithm Results:	difference	% diff.
4	646.65	635.9	10.75	1.690517
6	1055.2	1038.95	16.25	1.564079
8	1487.2	1462.55	24.65	1.685412
10	1936.05	1909.75	26.3	1.377144

**Table 9:** Average results from different number of clusters.





**Figure 12:** Different number of clusters for several code tests.

## 5. Conclusion and Future Work

The cluster-sequencing problem, recognized as one of the optimization problems, involves dealing with groups of damaged bridges that need to be repaired. The goal is to sequence the cluster repairs in a way that decrease the network delay. Therefore, the focus was using efficient optimization methods to repair cluster sequences. The solution methods applied to the cluster-sequencing problem deal with the improvement of the whole transportation network performance. And our approach involves working with the results obtained from the whole transportation network system.

Because of limited resources, damaged bridges cannot be repaired separately. Therefore, there are groups, or clusters, of damaged bridges in the transportation network. Clusters that would result in the most relief to the network delay take high priority. The optimal repair-sequence result is achieved by reaching the baseline stage.

The cluster-sequencing problem was solved by different optimization methods. The solution methods vary in their ways of handling the problem. Each solution method tracks the repair stages to end at the baseline delay. Furthermore, the applied optimization algorithms are ranging in their complexity from exponential to linear. It is proven that the solution to the cluster-sequencing problem is generally approachable from different optimization methods.

The first optimization method was a formulation of the cluster-sequencing problem was the “a dynamic programming approach to the modified travel salesman”\_ algorithm.

Then the problem was handled by the concept of one of the leading shortest-path algorithms. It is Dijkstra’s shortest path algorithm which sequences the cluster repairs. This algorithm gives optimal results.

The other approach to track the cluster-sequencing solution was achieved by establishing a heuristic method. This non-optimal algorithm could not guarantee the best results, but reached the solution more quickly.

Finally, a simulation method was developed to follow all the proposed repair states in all the stages. Based on the concept of binary numbers, this smart random assigns each state in each subsequent stage a different vector of binary numbers. It is a unique method from the point of estimating all the different states in each stage for the cluster repair sequence.

All the solution methods try to reach the case where there are no clusters to be repaired. This stage is achieved when all the clusters of damaged bridges are repaired. This final repair stage is the baseline stage —the equivalent of the pre-earthquake stage, when there was no damage to the transportation network. The solution methods will differ in their procedures to reach the baseline delay stage. For the solution methods to reach the baseline network delay, they follow different repair stages and sequences.

The ways that the cluster-sequencing problem was approached are important for the optimization field. The solution procedures concept can be implemented for many research problems. Just as our problem focuses on sequencing the clusters of damage bridges, some related problems that group together objects with similar characteristics can be considered to follow the same optimization procedures. Our approaches may also be used in problems that search for sequencing activities. The optimization field and other research fields in our lives have many problems that depend on different types of orders. So the idea established in this research of cluster-sequencing is considered to be efficient. In general, the different optimization problem characteristics will result in a range of changes in the application methods. Therefore, there will be a variation in the constraints and the objective functions of each separate application.

The cluster-sequencing problem is considered as a multistage problem. Therefore, the multistage problems may potentially use the same concept of defining the stages and states in each different size of cluster. The idea of using the stages and states plays a main role in the cluster-sequencing problem.

The unique application of the binary numbers towards the definition of the different cluster-repair sequences is an important concept. This definition can be involved in many uses of the binary numbers. Binary numbers specifies all the varying stages and states with a sole definition.

There are some possible extensions that our work is able to accommodate. One of the suggested extensions is that the research problem can deal with financial restrictions. These restrictions are used by assuming that a specific number of clusters only are to be considered in each construction or repair period. Furthermore, it could be possible to assume that all clusters that can be repaired within each budget period are repaired within that period. Applying the budget period assumes that the total construction or repair cost of each and all clusters is known. This can significantly reduce the number of groups that include bridges to be repaired.

There is another extension which is considered as a future work. Account for the variable importance of clusters. Essentially, the importance of a bridge can be determined by how many emergency centers are crossed by that cluster.

The clustering the damaged bridges requires more investigation. It is possible to cluster the damaged bridges by different clustering methods. One of the essential clustering procedures in TransCAD software is the area of influence method, a method that clusters the damage bridges depending on the ranking of different areas to a specific scale.

It is more realistic to examine different clustering methods for the same transportation network because, for many network, some clustering methods may be more suitable than others.

In general, it should be noted that the problem structures and solution approaches in the previous sections described the cluster-sequencing problem and tested the problem in different conditions. Furthermore, the clusters were defined as a group of different damaged bridges.

In the research problem, our approach was to deal with the results obtained from the whole transportation network system. Our concern is to show that the solution to the cluster-sequencing problem is generally approachable from several different optimization method concepts.

## References:

Ausiello, G., et al. (1999). *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties*. Heidelberg: Springer-Verlag.

Baesoz, N., and A.S. Kiremidjian. (1995). "Use of Geographic Information Systems for Bridge Prioritization." *Proceedings of the 5<sup>th</sup> International Conference on Seismic Zonation. French Association for Earthquake Engineering*. France: Earthquake Engineering Research Institute.

Baesoz, N., and A.S. Kiremidjian. (1997). "Damage to Bridges from the Northridge Earthquake and Its Consequences on Highway System Performance." *Proceedings of the 7<sup>th</sup> U.S.-Japan Workshop on Earthquake Disaster Prevention for Lifeline Systems*. Seattle.

Bellman, R. (1957). *Dynamic Programming*. Princeton: Princeton University Press.

Bellman, R. (1962). "Dynamic Programming Treatment of the Traveling Salesman Problem." *Journal of the Association for Computing Machinery* 9: 61-63.

Ben-Akiva, M. (1985). "Dynamic Network Equilibrium Research." *Transportation Research, Part A* 19A: 429-431.

Chang, S., and N. Nojima. (1997). "Highway System Performance Measures and Economic Impact." *Proceedings of the 7<sup>th</sup> U.S.-Japan Workshop on Earthquake Disaster Prevention for Lifeline Systems*. Seattle.

Chang, S., and N. Nojima. (1998). "Measuring Lifeline System Performance: Highway Transportation Systems In Recent Earthquakes." *Proceedings of the 6<sup>th</sup> U.S. National Conference on Earthquake Engineering*. Seattle.

Chang, S., and N. Nojima. (2001). "Measuring Post-Disaster Transportation System Performance: The 1995 Kobe Earthquake in Comparative Perspective." *Transportation Research, Part A: Policy and Practice* 35: 475-494.

- Cherng, R.-H., and Y.K. Wen. (1994). "Reliability-Based Cost-Effective Retrofit of Highway Bridge Systems." Proceedings of the 5<sup>th</sup> U.S. National Conference on Earthquake Engineering. Earthquake Engineering Research Institute.
- Dantzig, G.B., D.R. Fulkerson, and S.M. Johnson. (1954). "Solution of a Large-Scale Traveling Salesman Problem." *Operations Research* 2: 393-410.
- Dantzig, G.B., D.R. Fulkerson, and S.M. Johnson. (1985). *On a Linear Programming-Combinatorial Approach to the Traveling Salesman Problem*. Santa Monica: The Rand Corporation.
- David, S.K. (1991). *Dynamic Programming: A Practical Introduction*. New York: E. Horwood.
- Deakin, E. (1991). "Transportation Impacts of the 1989 Loma Prieta Earthquake: The Bay Area Bridge Closure." Working Paper. Berkeley: The University of California, Transportation Center.
- Desrosiers, J., Y. Dumas, and F. Soumis. (1986). "A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows." *American Journal of Mathematical and Management Sciences* 6: 301-325.
- Doohee, N., et al. (1999). "Analysis of the Impacts of Freeway Reconstruction Closure in Urban Areas." *Transportation Research Record* 161.1654: 161-170.
- Eash, R.W., B.N. Janson, and D.E. Boyce. (1979). "Equilibrium Trip Assignment: Advantages and Implications for Practice." *Proceedings of the 58<sup>th</sup> Annual Meeting of the Transportation Research Board*.
- Held, M., and R.M. Karp. (1962). "A Dynamic Programming Approach to Sequencing Problems." *Journal of Society of Industrial Math* 10.1: 196-210.
- Kameda, H., et al. (1990). "Effects of the 1989 Loma Prieta Earthquake on the Bay Area Transportation Systems." *Proceedings of the 8<sup>th</sup> Japan Earthquake Engineering Symposium, Vol. 2*.
- Karp, M.R., and M. Held. (1970). "The Traveling Salesman Problem and Minimum Spanning Tree." *Operations Research* 18: 1138-1162.



Kiyota, M., U. Vandebona, and H. Tanoue. (1999). "Multistage Optimization of Reconstruction Sequence of Highways." *Journal of Transportation Engineering* 125.5: 456-462.

Lawler, E.L. (1985). *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*. New York : Wiley.

Loma Prieta Earthquake, San Francisco, California October 17, 1989. (n.d.). Retrieved July 11, 2002, from [http://nisee.berkeley.edu/loma\\_prieta/loma\\_prieta.html](http://nisee.berkeley.edu/loma_prieta/loma_prieta.html).

Malandraki, C., and R.B. Dial. (1996). "A Restricted Dynamic Programming Heuristic Algorithm for the Time Dependent Traveling Salesman Problem." *European Journal of Operational Research* 90.1: 45-55.

Monroe, L., and A. Frank. (1968). "Optimal Construction Staging by Dynamic Programming." *Journal of the Highway Division*. The American Society of Civil Engineers.

Nemhauser, G.L. (1966). *Introduction to Dynamic Programming*. New York: Wiley.

Nigg, J.M. (1966). "Anticipated Business Disruption Effects Due to Earthquake-Induced Lifeline Interruptions." *Post-Earthquake Rehabilitation and Reconstruction*. F.Y. Cheng and Y.Y. Wang.

Norman, J.M. (1972). *Heuristic Procedures in Dynamic Programming*. Manchester: Manchester University Press.

Ortazar, J., and L.G. Willumsen. (2001). *Modeling Transport*, 3<sup>rd</sup> edition. Chichester: Wiley.

Papacostas, C.S., and Prevedouros, P.D. (1993). *Transportation Engineering and Planning*, 2<sup>nd</sup> edition. Englewood Cliffs: Prentice Hall.

Psaraftis, H. (1978). "A Dynamic Programming Approach to the Aircraft Sequencing Problem." *Massachusetts Institute of Technology*. Flight Transportation Laboratory. Report: R78-4.

Reinelt, G. (1994). *The Traveling Salesman: Computational Solutions for TSP Applications*. Berlin: Springer-Verlag.

- Sheffi, Y. (1985). *Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Methods*. London: Prentice-Hall.
- Shinozuka, M. (1999). *Effect of Earthquakes on Urban Highway Infrastructure Productivity, Vol. I*. Report to the National Science Foundation.
- Sungbin, C., et al. (2000). "Analyzing Transportation Reconstruction Network Strategies: A Full Cost Approach." *Review of Urban and Regional Development Studies* 12.3: 212-227.
- Thomas, R. (1991). *Traffic Assignment Techniques*. Brookfield: Avebury Technical.
- Tomofumi, N., and S. Hideki. (1997). "System to Assess Socio-Economic Effects of Earthquake Disaster." *Proceedings of the 7<sup>th</sup> U.S.-Japan Workshop on Earthquake Disaster Prevention for Lifeline Systems*. Seattle.
- Toth, P., and D. Vigo. (2002). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics.
- Wakabayashi, H., and H. Kameda. (1992). "Network Performance of Highway Systems under Earthquake Effects: A Case Study of the 1989 Loma Prieta Earthquake." *Proceedings of the 5<sup>th</sup> U.S.-Japan Workshop on Earthquake Disaster Prevention for Lifeline Systems*. Pages 215-232. Seattle.
- Werner, S., and J. Cooper. (1991). "Transportation Highways." *A Guide to Post-Earthquake Investigations of Lifelines*.
- Winston, W.L. (1994). *Operations Research: Applications and Algorithms*, 3<sup>rd</sup> Edition. London: Duxbury Press.

## Appendix A: Area of Study

The study area for our research problem will be applies to the San Francisco Bay Area transportation network. See Figure 13. It is after an earthquake magnitude 6.5 on the transportation system of an urban area.

Our research problem will sequence the clusters specified to the optimal repair sequence. As a result the total transportation network delay will decrease and network performance increase.

As a general look on the study area it is useful to get an idea about how bridges are clustered. One of the procedures used is that the bridges are clustered depending on a method used in a research project. It is a technique implemented by the project group (2000). The clusters for bridges are defined by finding a minimum span tree between the damage bridges. Then the distances are arranged ascending from a specified bridge. Next an algorithm used to sort the distances to find the discontinuous value. Each break between bridge distances means that it is a cluster. All the data required from the study area are available. For example we know the volume, distance, and capacity for all links in the transportation network.

Furthermore, other formation of bridge cluster methods can be used with different procedures and purpose. It is such the area of influence method. When there is an important location take more weight.

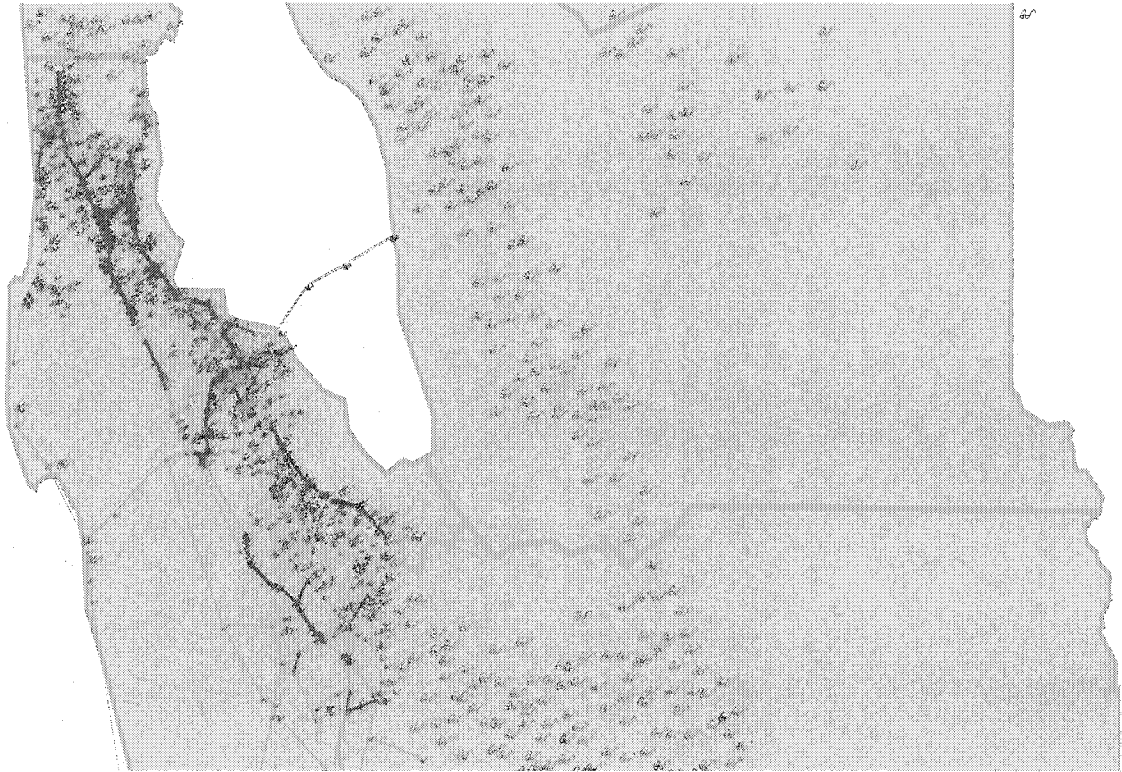
Table 10 show closed links numbers and number of cluster they belong. It is for an example of 3 clusters. Each closed link is connected to a damage bridge.

Closed Link Number	Cluster Number	Closed Link Number	Cluster Number
1276	1	1609	
1275	1	967	2
1314	1	1545	2
1258	1	1947	2
1293	1	1543	2
1292	1	1544	2
987	1	1274	2
1347	1	1996	2
1286	1	1941	2
1290	1	1995	2
1291	1	1903	2
115	1	64	3
703	1	1921	3
113	1	1988	3
1137	1	1980	3
1136	1	2010	3
2242	2	1866	3
1626	2	2011	3
1611	2	1864	3
1611	2	2016	3
1612	2	1983	3

**Table 10:** Closed links number for each cluster.



**Figure 13:** San Francisco Bay Area transportation network



**Figure 14:** Bay Area Clusters



**Figure 15:** The specified area of study in the Bay Area

## Appendix B: TransCAD Results:

### Results of 3 clusters Nothing repaired:

Iteration	Step	Starting Procedure Traffic Assignment			
		Relative Gap	Max. Flow Chang	RMSE	% RMSE
1	0.484280	0.603035	1109.992248	86.10	674.16
2	0.097545	0.057908	207.324556	11.65	91.24
3	0.077582	0.042511	163.361687	8.48	66.55
4	0.051327	0.025727	102.368726	5.36	42.09
5	0.040192	0.018368	84.981762	4.59	36.06
6	0.016615	0.008183	30.838870	1.62	12.72

#### INPUT FILES

Network: C:\Documents and Settings\Fahad\My Documents\Bay Area files  
(TransCAD & GIS)\3 clusters Nothing repaired.net

Demand Table: C:\Documents and Settings\Fahad\My Documents\Bay Area files  
(TransCAD & GIS)\Bay Area files (TransCAD & GIS)\OD matrix.mtx

#### OUTPUT FILES

Flow Table: C:\Documents and Settings\Fahad\My Documents\Bay Area files  
(TransCAD & GIS)\TA for 3 Clusters Nothing Repaired (OD index NEW)(Srin  
Method in 8-21).bin

#### LINK FIELDS

Cost : TIME  
Capacity : CAPACITY

#### OD DEMAND

OD Pairs : 441  
Non zero OD Pairs : 401  
Demand : 7829.02  
Intranodal Demand : 770.37

#### PARAMETERS



Method : User Equilibrium  
 Maximum Iterations : 50  
 Iterations : 7  
 Conv. Criteria : 0.01

#### Running Results

Relative Gap : 0.01  
 RMSE : 1.62  
 % RMSE : 12.72  
 Max Flow Change : 30.84  
 Equilibrium reached : Yes  
 Total V-Time-T : 83126.81  
 Total V-Dist-T : 27808.05

#### Results of 3 clusters ALL repaired:

Starting Procedure Traffic Assignment					
Iteration	Step	Relative Gap	Max. Flow Change	RMSE	% RMSE
1	0.490662	0.665191	1066.782819	85.23	867.59
2	0.019660	0.016219	22.828659	1.75	17.87
3	0.012301	0.009931	26.728724	1.29	13.16

#### INPUT FILES

Network: C:\Documents and Settings\Fahad\My Documents\Bay Area files  
 (TransCAD & GIS)\3 Clusters All Repaired.net  
 Demand Table: C:\Documents and Settings\Fahad\My Documents\Bay  
 Area files (TransCAD & GIS)\Bay Area files (TransCAD & GIS)\OD matrix.mtx

#### OUTPUT FILES

Flow Table: C:\Documents and Settings\Fahad\My Documents\Bay Area  
 files (TransCAD & GIS)\TA for 3 Clusters All Repaired(OD matrix index is  
 NEW).bin

### LINK FIELDS

---

Cost : TIME  
Capacity : CAPACITY

### OD DEMAND

---

OD Pairs : 441  
Non zero OD Pairs : 401  
Demand : 7829.02  
Intranodal Demand : 770.37

### PARAMETERS

---

Method : User Equilibrium  
Maximum Iterations : 50  
Iterations : 4  
Conv. Criteria : 0.01

### Running Results

---

Relative Gap : 0.01  
RMSE : 1.29  
% RMSE : 13.16  
Max Flow Change : 26.73  
Equilibrium reached : Yes  
Total V-Time-T : 68352.74  
Total V-Dist-T : 20847.19

## Appendix C: C++ Codes

We used c++ programming language to solve the cluster-sequencing problem. It handles the two methods of solutions. They are the Dijkstra algorithm and the heuristic approach algorithm. Both of them give results which are comparable and analyzed.

```
// Cluster ver 3.0 // Heuristic Approach + Shortest Path algorithms

#include<iostream>
#include<iomanip>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

using namespace std;

const int INFINITY = 9999;
const int MAX_CL = 40;
int graph[MAX_CL][MAX_CL];
int cost [MAX_CL][MAX_CL];
bool visited [MAX_CL];
int parent [MAX_CL], D [MAX_CL];

int ncl; // number of cluster

int assignedRow=0;
int assignedCol=0;

// Function prototypes

void initialize(int n); //inititalize a graph on size n
void display ();
```

```

void greedy ();
void printSequence (int);
void Dijkstra ();
int myPower(int, int);

int UE(int i, int j, int max, int min); // a function to estimate the cost at (i,j).
                                         // the cost is between min and max.

////////////////////////////////////
// main function
////////////////////////////////////

int main ()
{
    //int POWER;
    //int ncluster;
    int n;
    char c = 'a';

    srand( (unsigned)time( NULL ) ); // If it is used it will seeding the random generator
                                     for //UE

    while (c != 'q')
    {
        cout<<setw(20) << "Enter the size of the graph: ";
        cin>>n;
        //cin>>ncluster;
        //n=POWER(2,ncluster);
        n = myPower(2,n); //n= 2^n

        initialize(n); // build the graph

        display(); // display the graph
        greedy(); // run the greedy algorithm

        cin>>c;

        dijkstra(); // run Dijkstra shortest path algorithm

        cin>>c;
    }
}

```

```

    return 0;
} //end of main

```

```

////////////////////////////////////
// Initialize
////////////////////////////////////

void initialize(int n)
{
    int i,j;
    int max = 100; // max cost
    int min = 10; // min cost

    ncl = n;

    for (i=0; i < n; i++)
    {
        for (j=0; j<n; j++)
        {
            cost[i][j] = UE(i,j, max, min); // call the cost function
            if (cost[i][j] == INFINITY)
                graph[i][j] = 0;
            else
                graph[i][j] = 1;
        }
    }
}

////////////////////////////////////
// Display()
////////////////////////////////////

void display()
{
    int i,j;

    cout<<"Here is a graph of "<<ncl<<" nodes:"<<endl<<endl;
    cout<<"nodes ";
    for(j=0; j<ncl ; j++)

```

```

        cout<<setw(4)<<j;
    cout<<endl;

    for(i=0; i< ncl; i++)
    {
        cout<<setw(4)<<i<<" ";
        for(j=0; j<ncl; j++)
        {
            if (cost[i][j] < INFINITY)
                cout<<setw(4)<<cost[i][j];
            else
                cout<<setw(4)<<"-";

        }
        cout<<endl;
    }
}

////////////////////////////////////
// Heuristic approach algorithm
////////////////////////////////////

void greedy() // version 2
{
    int i,j,k, cur, best_cost, min_j, total = 0, sum = 0;

    for (i=0; i<ncl ; i++)
        visited[i] = false;

    cout<<"\n *** Heuristic Results: *** \n The sequence is: "<<endl;
    cur = 0;

    for(k=0; k< ncl; k++)
    {
        best_cost = INFINITY;
        visited[cur] = true;
        cout << "node_" << cur << endl;
        for (j = 0; j < ncl; j++)

```

```

        if (cost[cur][j] < best_cost && !visited[j])
        {
            best_cost = cost[cur][j];
            min_j = j;
        }

        cur = min_j;
        sum += best_cost;
        if (min_j == ncl-1) // this means we found the goal
        {
            total = sum;
            k = ncl; // to stop the main loop
        }
    }

    cout << "node_" << cur << endl;

    cout<<"\nTotal cost = "<<total<<endl;
}

/////////////////////////////////////////////////////////////////
// Shortest Path Algorithm
/////////////////////////////////////////////////////////////////

void dijkstra()
{
    int i,j, v, k, Dk, total = 0;
    int n = ncl;

    cout<<"\n *** Shortest Path Results: ***\nThe sequence is: "<<endl;

    for (i=0; i<n ; i++)
        visited[i] = false;

    for (i=0; i < n; i++)
    {
        D[i] = cost[0][i];
        parent[i] = 0;
        if(n-1 == i)
            total = D[i];
    }
}

```

```

    }

    visited[0] = true; //add 0 to S = visited
    parent[0] = -1;
    D[0] = 0;

    for(i=0; i< n-1; i++) // repeated n-1 times
    {
        Dk = INFINITY;
        for (j = 0; j < n; j++) // to pick the min k
            if (D[j] < Dk && !visited[j])
            {
                Dk = D[j];

                k = j;
            }
        visited[k] = true; // add the min k to S = visited

        for(v=0; v< n; v++)
            if (!visited[v])
                if (D[v] > D[k]+cost[k][v])
                {
                    D[v] = D[k]+cost[k][v];
                    parent[v] = k; // claim the parenthood
                    if (v == n-1) // check if last node (target)
                        total = D[v];
                }
    } // end of for loop = end of algorithm.

    // printing the sequence...

    printSequence(n-1);

    cout<<"\nTotal cost = "<<total<<endl;
}

////////////////////////////////////
//  printSequence;
////////////////////////////////////
void printSequence(int k)
{
    if (parent[k] >= 0) // not the root
        printSequence(parent[k]);
}

```



```

    cout<<"node_"<<k<<endl;
}

////////////////////////////////////
// my Power
////////////////////////////////////

int myPower(int a, int k) // to compute result = a^k

{
    int i, result=1;

    for (i=1; i <=k; i++)
        result = result * a;
    return (result);
}

////////////////////////////////////
// Time
////////////////////////////////////

int UE(int i, int j, int max, int min)

{
    //any number of clusters

    (11...1)(011...1)(101...1)...(100...1)(010...1)...(000...0)

    int a[n][n]={
                                                    // (11 ... 1)
                                                    // (011 ... 1)
                                                    // (101 ... 1)
                                                    // (100 ... 1)
                                                    // (010 ... 1)
                                                    } // (000 ... 0)

```

```
if (i == j)
    return (INFINITY); // no selfloop
else
    return (a[i][j]);
}
```

## Appendix D: Codes for testing the simulating method

```

#include<iostream>
#include<iomanip>
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

using namespace std;

int Min and Max costs

//////////

const int MAX_CLS = 10;
const int MAX_N = 1025; // 2^10
int graph[MAX_N][MAX_N];
int cost [MAX_N][MAX_N];
bool visited [MAX_N];
int parent[MAX_N], D[MAX_N];
int tt[MAX_N]; // tags tables;

int ncl; // number of columns in the matrix = number of nodes
int cls; // nubmer of clusters

int assignedRow=0;
int assignedCol=0;

int seed; // for prandom

int test;

const int r = (A * seed + C) % M

// function prototypes

void initialize(int n); //inititalize a graph on size n
void display();

```

```

void greedy();
void printSequence(int);
void dijkstra();
int myPower(int, int);

int prandom(int, int); //pseudo random
int tag (int); // finds the tag of node i
int wt(int); // finds the wieght of node i = number of 1's in i as binary

int UE(int i, int j); // a function to estimate the cost at (i,j).

/////////////////////////////////////////////////////////////////
// main function
/////////////////////////////////////////////////////////////////

int main()
{
    //int POWER;
    //int ncluster;
    int n;
    char c = 'a';

    srand( (unsigned)time( NULL ) ); // seeding the random generator for UE

    while (c != 'q')
    {
        cout<<setw(20)<< "Enter number of clusters (max 10): ";
        cin>>cls;
        n = myPower(2,cls); //n= 2^n
        cout<<setw(20)<< "Enter the test number (seed between 1-5000): ";
        cin>> seed;
        test = seed;
        seed = seed + 101;

        initialize(n); // build the graph

        display(); // display the graph
        greedy(); // run the greedy algorithm

        cin>>c;
    }
}

```

```

        dijkstra(); // run Dijkstra shortest path algorithm
        cout<<setw(20) << "When you are done, hit q to quit";
        cin>>c;
    }
    return 0;
} //end of main

/////////////////////////////////////////////////////////////////
// Initialize
/////////////////////////////////////////////////////////////////

void initialize(int n)
{
    int i,j;
    ncl = n;

    for (i=0; i < n; i++) //reset the graph to 0. tag(i) will set the 1's.

        for (j=0; j < n; j++)

            graph[i][j] = 0;

    tt[0] = Max; // the cost at node 0 is the time of maximum damage.

    for (i=1; i < n; i++)
        tt[i] = tag(i);

    tt[n-1] = Min; // the cost at last node is the minimum.

    for (i=0; i < n; i++)
    {
        for (j=0; j<n; j++)
        {
            cost[i][j] = UE(i,j); // call the cost function
        }
    }
}

/////////////////////////////////////////////////////////////////

```

```

// Display()
////////////////////////////////////

void display()
{
    int i,j;

    cout<<"\n * * * Graph of size: "<<ncl<<" nodes,   Test #: "<< test <<endl<<endl;
    cout<<"nodes ";
    for(j=0; j<ncl ; j++)
        cout<<setw(4)<<j;
    cout<<endl;

    for(i=0; i< ncl; i++)
    {
        cout<<setw(4)<<i<<": ";

        for(j=0; j<ncl; j++)
        {
            if (cost[i][j] < INFINITY)
                cout<<setw(4)<<cost[i][j];
            else
                cout<<setw(4)<<"-";

        }
        cout<<endl;
    }
}

////////////////////////////////////
// Heuristic approach algorithm
////////////////////////////////////

void greedy()
{
    int i,j,k, cur, best_cost, min_j, total = 0, sum = 0;

    for (i=0; i<ncl ; i++)

```

```

        visited[i] = false;

    cout<<"\n *** Heuristic Approach Results: ***\nThe sequence is: "<<endl;
    cur = 0;
    for(k=0; k< ncl; k++)
    {
        best_cost = INFINITY;
        visited[cur] = true;
        cout<<"node_"<<cur<<endl;

        for (j = 0; j < ncl; j++)
            if (cost[cur][j] < best_cost && !visited[j])
            {
                best_cost = cost[cur][j];
                min_j = j;
            }

        cur = min_j;

        sum += best_cost;
        if (min_j == ncl-1)
        {
            total = sum;
            k = ncl;    // to stop the main loop

        }
    }
    cout << "node_" << cur << endl;
    cout<<"\nTotal cost = "<<total<<endl;
}

```

```

////////////////////////////////////
// Shortest path Algorithm
////////////////////////////////////

```

```

void dijkstra()
{
    int i,j, v, g, Dg, total = 0;
    int n = ncl;

    cout<<"\n *** Shortest path Results: ***\nThe sequence is: "<<endl;

    for (i=0; i<n ; i++)

```

```

        visited[i] = false;

    for (i=0; i < n; i++)
    {
        D[i] = cost[0][i];

        parent[i] = 0;
        if(n-1 == i)
            total = D[i];
    }

    visited[0] = true; //add 0 to S = visited
    parent[0] = -1;
    D[0] = 0;

    for(i=0; i<n-1; i++) // repeated n-1 times
    {
        Dg = INFINITY;
        for (j = 0; j < n; j++) // to pick the min g
            if (D[j] < Dg && !visited[j])
            {
                Dg = D[j];

                g = j;
            }
        visited[g] = true; // add the min g to S = visited

        for(v=0; v<n; v++)
            if (!visited[v])
                if (D[v] > D[g]+cost[g][v])
                {
                    D[v] = D[g]+cost[g][v];
                    parent[v] = g; // claim the parenthood
                    if (v == n-1) // check if last node (target)
                        total = D[v];
                }
    } // end of for loop = end of alg.

    // printing the sequence...

    printSequence(n-1);

    cout<<"\nTotal cost = "<<total<<endl;

```



```

}

/////////////////////////////////////////////////////////////////
//  printSequence;
/////////////////////////////////////////////////////////////////
void printSequence(int k)
{
    if (parent[k] >= 0) // not the root
        printSequence(parent[k]);

    cout<<"node_"<<k<<endl;
}

/////////////////////////////////////////////////////////////////
// my Power
/////////////////////////////////////////////////////////////////

int myPower(int a, int k) // to compute result = a^k
{
    int i, result=1;
    for (i=1; i <=k; i++)
        result = result * a;
    return (result);
}

/////////////////////////////////////////////////////////////////
// P random - pseudo random between a, b (inclusive)
/////////////////////////////////////////////////////////////////

int prandom(int a, int b) // uses and updates global seed
{
    int r;

    r = (A * seed + C) M;
    seed = r;
    if (b > a)
        return (a + (r * (b - a + 1 M) ));
    else
        return a;
}

```

```

////////////////////////////////////
// tag
////////////////////////////////////

int tag (int j) // finds the tag of node i
{
    int x, min = INFINITY, f = 1;
    int s, a, b, budget;
    x = j;

    while (f <= x)
    {
        if ( (x % (2 * f)) != 0 )
        {
            x = x - f;
            graph[j-f][j] = 1;
            if ( min > tt[j-f] )
                min = tt[j-f];
        }

        f *= 2;
    }

    // min is found

    s = cls - wt(j) + 1;

    budget = min - Min; // current_min - abs_min

    b = min; // min is now the upper pound. next cost must be less.

    a = min - (2* budget / s) + s - 1 ;

    return prandom(a, b); // next cost is a random between a and b
}

```

```

////////////////////////////////////
// Wieght
////////////////////////////////////

int wt(int x)  // finds the wieght of node i = number of 1's in i as binary
{
    if (x == 0)
        return 0;
    else
        return ( (x % 2) + wt(x/2) );
}

////////////////////////////////////
// Travel time
////////////////////////////////////

int UE(int i, int j)
{
    if ( graph[i][j] == 1 ) // there is a link
        return tt[j];
    else // no link
        return (INFINITY);
}

```