HEURISTIC ALGORITHM FOR
AN INTERNET MESH NETWORK DESIGN PROBLEM

BY

PANUPONG VONGSARIYAVANICH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING IN LOGISTICS AND SUPPLY CHAIN SYSTEMS ENGINEERING SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY THAMMASAT UNIVERSITY ACADEMIC YEAR 2014
HEURISTIC ALGORITHM FOR
AN INTERNET MESH NETWORK DESIGN PROBLEM

BY

PANUPONG VONGSARIYAVANICH

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF MASTER OF ENGINEERING
IN LOGISTICS AND SUPPLY CHAIN SYSTEMS ENGINEERING
SIRINDHORN INTERNATIONAL INSTITUTE OF TECHNOLOGY
THAMMASAT UNIVERSITY
ACADEMIC YEAR 2014
HEURISTIC ALGORITHM FOR
AN INTERNET MESH NETWORK DESIGN PROBLEM

A Thesis Presented

By

PANUPONG VONGSARIYAVANICH

Submitted to
Sirindhorn International Institute of Technology
Thammasat University
In partial fulfillment of the requirements for the degree of
MASTER OF ENGINEERING IN
LOGISTICS AND SUPPLY CHAIN SYSTEMS ENGINEERING

Approved as to style and content by

Advisor and Chairperson of Thesis Committee
(Assoc. Prof. Navee Chiadamrong, Ph.D.)

Co-Advisor
(Panitan Kewcharoenwong, Ph.D.)

Committee Member and
Chairperson of Examination Committee
(Assoc. Prof. Pisal Yenradee, D.Eng.)

Committee Member
(Asst. Prof. Teeradej Wuttipornpun, Ph.D.)

JULY 2015
Abstract

HEURISTIC ALGORITHM FOR
AN INTERNET MESH NETWORK DESIGN PROBLEM

by

Panupong Vongsariyavanich

B.Eng (Industrial Engineering) Sirindhorn International Institute of Technology,
Thammasat University, 2011

This study presents the algorithm based on Local Search heuristic used to solve the internet mesh network design problem. The network in this study is a mesh network, formulated as a Mixed Integer Programming (MIP) model. The mathematical model is tested with a branch and cut algorithm using CPLEX concert technology with C++. The proposed Local Search algorithm consists of 3 stages where each consecutive stage aims for progressive enhancement over the previous one. First, LS1 is the basic iterated Local Search algorithm. This stage introduces many insights on the solution generation and evaluation process of the algorithm. Then, LS2 aims for improvement over LS1, which utilizes the Harmony Search algorithm concept and idea. At last stage, LS3 derives an implementation from the objective function to instruct the solution generation process more efficiently. By analysis and comparison though many tested instances and scenarios, the results obtained show that LS3 yields the best performance from out of the 3 Local Search algorithm and can solve a large problem size that a branch and cut algorithm cannot manage.
Table of Contents

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Signature Page</td>
<td>i</td>
</tr>
<tr>
<td></td>
<td>Abstract</td>
<td>ii</td>
</tr>
<tr>
<td></td>
<td>Table of Contents</td>
<td>iii</td>
</tr>
<tr>
<td></td>
<td>List of Figures</td>
<td>v</td>
</tr>
<tr>
<td></td>
<td>List of Tables</td>
<td>vi</td>
</tr>
<tr>
<td>1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Problem statement</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Objective of thesis</td>
<td>2</td>
</tr>
<tr>
<td>1.3</td>
<td>Overview of thesis</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Literature Review</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Problem Definition</td>
<td>6</td>
</tr>
<tr>
<td>3.1</td>
<td>Network overview</td>
<td>6</td>
</tr>
<tr>
<td>3.2</td>
<td>Network characteristics</td>
<td>7</td>
</tr>
<tr>
<td>3.3</td>
<td>Input data</td>
<td>8</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Distance data</td>
<td>8</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Capacity data</td>
<td>10</td>
</tr>
<tr>
<td>3.3.3</td>
<td>Cost data</td>
<td>11</td>
</tr>
<tr>
<td>3.3.4</td>
<td>Service data</td>
<td>13</td>
</tr>
<tr>
<td>3.4</td>
<td>Output data</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>Model Formulation</td>
<td>15</td>
</tr>
<tr>
<td>4.1</td>
<td>Notations</td>
<td>15</td>
</tr>
<tr>
<td>4.2</td>
<td>Parameters</td>
<td>15</td>
</tr>
<tr>
<td>4.3</td>
<td>Decision variables</td>
<td>16</td>
</tr>
<tr>
<td>4.4</td>
<td>Mathematical model equation</td>
<td>16</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Objective function</td>
<td>16</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Constraints</td>
<td>17</td>
</tr>
<tr>
<td>5</td>
<td>Methodology</td>
<td>20</td>
</tr>
<tr>
<td>Chapter</td>
<td>Title</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>5.1</td>
<td>Branch and cut algorithm</td>
<td>20</td>
</tr>
<tr>
<td>5.2</td>
<td>Proposed heuristic algorithm</td>
<td>20</td>
</tr>
<tr>
<td>5.2.1</td>
<td>Solution representation</td>
<td>21</td>
</tr>
<tr>
<td>5.2.2</td>
<td>Solution evaluating process</td>
<td>23</td>
</tr>
<tr>
<td>5.3</td>
<td>LS1 algorithm</td>
<td>23</td>
</tr>
<tr>
<td>5.3.1</td>
<td>LS1 solution generating process</td>
<td>24</td>
</tr>
<tr>
<td>5.4</td>
<td>LS2 algorithm</td>
<td>25</td>
</tr>
<tr>
<td>5.4.1</td>
<td>HS algorithm</td>
<td>25</td>
</tr>
<tr>
<td>5.4.2</td>
<td>LS2 solution generating process</td>
<td>28</td>
</tr>
<tr>
<td>5.5</td>
<td>LS3 algorithm</td>
<td>28</td>
</tr>
<tr>
<td>5.5.1</td>
<td>LS3 solution generating process</td>
<td>28</td>
</tr>
<tr>
<td>6</td>
<td>Design of Experiments</td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td>Results and Discussions</td>
<td>32</td>
</tr>
<tr>
<td>7.1</td>
<td>Comparison to CPLEX</td>
<td>32</td>
</tr>
<tr>
<td>7.1.1</td>
<td>Gap comparison</td>
<td>32</td>
</tr>
<tr>
<td>7.1.2</td>
<td>Runtime comparison</td>
<td>35</td>
</tr>
<tr>
<td>7.2</td>
<td>Comparison to LS3</td>
<td>38</td>
</tr>
<tr>
<td>8</td>
<td>Conclusion and Recommendation</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>References</td>
<td>42</td>
</tr>
<tr>
<td></td>
<td>Appendix A: Proposed Algorithm Code</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Appendix B: Example of Data and Network Result Text File</td>
<td>65</td>
</tr>
<tr>
<td></td>
<td>Appendix C: Objective Value and Runtime Result Table</td>
<td>72</td>
</tr>
</tbody>
</table>
## List of Figures

<table>
<thead>
<tr>
<th>Figures</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>WOBAN architecture</td>
<td>4</td>
</tr>
<tr>
<td>3.1</td>
<td>Network topology</td>
<td>6</td>
</tr>
<tr>
<td>5.1</td>
<td>LS1 algorithm flowchart</td>
<td>24</td>
</tr>
<tr>
<td>5.2</td>
<td>2 Behaviors of solution generating process of LS1</td>
<td>25</td>
</tr>
<tr>
<td>5.3</td>
<td>LS2 algorithm flowchart</td>
<td>26</td>
</tr>
<tr>
<td>5.4</td>
<td>3 Behaviors of Improvise Search</td>
<td>27</td>
</tr>
<tr>
<td>7.1</td>
<td>Percentage gap of LSx as compared to CPLEX</td>
<td>33</td>
</tr>
<tr>
<td>7.2</td>
<td>Average percentage gap of LSx as compared to CPLEX</td>
<td>34</td>
</tr>
<tr>
<td>7.3</td>
<td>Standard deviation gap of LSx</td>
<td>35</td>
</tr>
<tr>
<td>7.4</td>
<td>Runtime Comparison</td>
<td>36</td>
</tr>
<tr>
<td>7.5</td>
<td>Average of runtime comparison</td>
<td>37</td>
</tr>
<tr>
<td>7.6</td>
<td>Standard deviation of runtime comparison</td>
<td>37</td>
</tr>
<tr>
<td>7.7</td>
<td>Percentage gap of LSs as compared to LS3</td>
<td>39</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Tables</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.1</td>
<td>Example of xyz coordination data</td>
</tr>
<tr>
<td>3.2</td>
<td>Example of rectilinear distance data (meter)</td>
</tr>
<tr>
<td>3.3</td>
<td>Example of euclidean distance data (meter)</td>
</tr>
<tr>
<td>3.4</td>
<td>Example of capacity of wire connection data (unit)</td>
</tr>
<tr>
<td>3.5</td>
<td>Example of capacity of wireless connection data (unit)</td>
</tr>
<tr>
<td>3.6</td>
<td>Example of cost of locating access point data (Baht)</td>
</tr>
<tr>
<td>3.7</td>
<td>Example of cost of installing physical wire data (Baht)</td>
</tr>
<tr>
<td>3.8</td>
<td>Example of relative demand data (unit)</td>
</tr>
<tr>
<td>5.1</td>
<td>Example of solution matrix</td>
</tr>
<tr>
<td>5.2</td>
<td>Example of AP location vector</td>
</tr>
<tr>
<td>5.3</td>
<td>Example of transmission matrix</td>
</tr>
<tr>
<td>6.1</td>
<td>Data structure A and B</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

Network flow and network design models have been in literature for many decades. They are common and widely used in transportation and telecommunication industries. To solve these networks efficiently, a specific implementation is needed because each network contains its own characteristics. Understanding throughout the network can sufficiently improve the quality of the algorithm due to the full control over the algorithm being implemented.

The network design in this problem can be considered to be the same as transportation problems. Meaning that, some network design equations can be used to solve this similar kind of problem. The network model is derived from one of the classic models, namely Multi-Commodity Network Flow model and applied into the communication network design problem. Some of the parameters in the model need to be changed to match with this problem. For example, the cost of transportation and the distance between each distribution center can be derived as the cost of data transmission and the distance between each relay device, respectively.

1.1 Problem statement

This thesis focuses on developing an algorithm to solve a particular network. Existing algorithms that have been used in optimization consume too much memory and are not efficient to solve big problem size. Moreover, each network design problem contains its own particular characteristics. Therefore, a specific algorithm is needed to solve this underlying network design problem.

The network in this study is derived into a communication network. Since, the practical communication network involves many different technical knowledge, there are many assumptions used in this study for simplification purpose and keeping the focus on the algorithm aspect.
1.2 Objectives of thesis

The main content of this thesis is to develop an algorithm based on meta-heuristic to efficiently solve the underlying network design problem. The objectives are as follows:

1. Develop a Mixed Integer Programming (MIP) model for the underlying network.
2. Analyze and evaluate the behavior and results obtained from the derived algorithm in order to give some useful insights.

1.3 Overview of thesis

This thesis is organized into 8 chapters. First, Chapter 1 introduces the network design problem as well as the objectives of the thesis. Chapter 2 includes review of the related literature. Chapter 3 gives the information about the underlying network. Then, the proposed network is formulated into mathematical model, which is discussed in chapter 4. Chapter 5 presents the methodology and the proposed heuristic algorithm. The design of the experiments will be discussed in chapter 6. The obtained results and discussions are includes in chapter 7. Finally, the conclusion and recommendation are addressed in chapter 8.
There are a number of researches in the area related to the internet network design problem. In the context of designing the network, each research contains unique network characteristics and requirements. Here, only research directly related to this topic is included.

Gzara and Erkut (2011) considered a telecommunications network, which allows multiple technologies. The multiple technologies allow two types of fibers wire: fibers laid in parallel and on consecutive fibers. If different technologies are used in the same path, a switch is required to transform the signal in order to make communication between two different technologies. Randazzo et al. (2001) also used this concept in their researches. But they only focused on using the technology of the wire, so there are two different fiber technologies to be considered.

Kewcharoenwong and Uster (2013) worked on an optical network. In optical network, the signal quality will decrease as it travels through many intermediate relay nodes. This causes physical impairment, distortion, cross-talk, dispersions, and power failure. Therefore, they applied the O/E/O technology to the relay node for improving the signal quality. The O/E/O will perform a signal regeneration, re-amplify, re-shape, and re-time. This leads to their problem context such that installed O/E/O at every node will be very expensive and inefficient.

Some researches’ objective is to minimize the cost of construction and operation of the network, other researches’ objective focuses on different objective functions. One interesting topic is to minimize the energy consumption of the network. Chowdhury et al. (2010) focused on developing the model to act as a guideline on designing an energy-aware of Wireless-Optical Broadband Access Network (WOBAN). Figure 2.1 is the WOBAN architecture. The structure as shown in Figure 2.1 is also similar to the mesh network where each node can connect to each other.
In the context of solution methodology, this network design-type is MIP, which is known to have an inherit complexity that requires a customized algorithm for solving large problem size. So and Liang (2009) used Bender’s decomposition technique to compute the optimal number of Relay Stations (RSs) and their corresponding placement as well as channel assignment in order to minimize the operation cost. Their proposed Bender’s decomposition utilizes the heuristic and objective-cut method to reduce the runtime.

Although the network in this study draws the network design into a communication network, it contains assumptions that make the underlying network intersect between practical communication and demand-supply network. Rodriguez-Martin and Salazar-Gonzalez (2010)’s network is the capacitated fixed-charge network design problem. It shares some common characteristics with the network in this study. Their study proposed a meta-heuristic technique called Local branching, which utilizes a general MIP solver to explore the neighborhoods.

A practical communication network involves many areas of expertise, such as the package transmit signal, the specification of the access point, the intersect signal frequency, which causes a signal to drop, the network operating over a time period. Bruno et al. (2011) stated that, in practical wireless mesh network (WMNs), gateways are subject to hard capacity limiting on the aggregate number of flows, which can
cause some gateways or intermediate mesh routers overloaded if the traffic is routed without considering those constraints as well as the traffic distribution.

Looking at different points of view of the internet network design, Barbosa and Gouvea (2012) focused on the Access Point (AP) design using a Genetic Algorithm. Their study presented a novel model to design the AP location, which maximizes the coverage area and the user connection.

The proposed algorithm in this study is the Local Search algorithm with Harmony Search concept. Harmony Search algorithm has been shown to achieve excellent results in wide range of optimization problems (Manjarres et al., 2013). In Harmony Search, there are parameters that control how the algorithm performs. Having tweaked these parameter settings, Harmony Search can also act as other heuristics algorithm concepts (discussed later in this study).

There are many researches used Harmony Search heuristic to solve the network design problem. Landa-Torres et al. (2013) proposed an approach based on Harmony Search algorithm. One of interesting aspects is the adaptation of the Harmony Search algorithm to grouping scheme and the improvisation operators driving the algorithm.
CHAPTER 3
PROBLEM DEFINITION

Each network has its own structure and operational requirements. These specifications make each network design problem distinct from each other. Therefore, understanding throughout the problem is required as a starting point. This chapter describes the presented network in this study.

3.1 Network overview

The network in this study is a mesh communication network on a 3 dimensional plane. One node in the network is the location of the main router (origin) whereas others nodes (destinations) are a candidate position to locate the access point. The origin node receives signals from the external source and transmits to all other nodes, which require a signal. This type of network is a single source/origin – many sinks/destinations network.

There are two potential connection types between nodes, either wire or wireless. Thus, the resulting network could be a pure LAN network, a pure WLAN network, or a hybrid network.

Figure 3.1: Network topology
Figure 3.1 illustrates the network topology in this study, noting that the actual network is on a 3 dimension. The triangle represents the main router whereas the circle represents the access point location on selected nodes. The solid line represents the wire connection whereas dash line represents the wireless connection. Each node can connect to other nodes like a mesh structure. The star shape represents a user location connects to the in-range access point. However, in this study, the signal frequency is not within the area of interest. Therefore, the dash circle and star shape in Figure 3.1 illustrate the demand attached to each access points, which are standardized into relative demand for simplification and algorithm focused purpose. Thus, the capacity is also standardized.

3.2 Network characteristics

The presented network has following structural and operational requirements:
1. There is only one type of connection (wire or wireless) between two nodes.
2. Wireless connection has a connection range constraint. Therefore, two access points can be connected using wireless connection if their associate euclidean distances are within the pre-specified distance.
3. All relative demand must be satisfied. In other words, all users currently accessing to each access point must receive enough signal.
4. The transmitting signal can use multiple paths.
5. The total transmission over a link cannot exceed its respective connection type capacity.
6. The wire connection between two nodes requires two access points to be located at both end nodes. Moreover, the actual wire must be installed prior to the signal transmission.
7. The wireless connection between two nodes also requires two access points to be located at both end nodes.
3.3 Input data

The minimum requirement of the input data can be classify into four categories: distance, capacity, cost, and service. For the purpose of explanation, the network of 5 nodes will be used as an example in order to show the representation of each data.

3.3.1 Distance data

The distance data category consists of xyz coordination of each node, the rectilinear distance and euclidean distance between each node, and the maximum distance for making wireless connection.

**XYZ coordination**

Table 3.1: Example of xyz coordination data

<table>
<thead>
<tr>
<th>Node</th>
<th>x</th>
<th>y</th>
<th>z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>7</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 3.1 illustrates the xyz coordination data. It is an \((N \times 3)\) matrix where \(N\) is a number of node and 3 is a \((x, y, z)\) coordination. For example, node 1 has \((0, 1, 0)\) coordination, node 2 has \((10, 0, 5)\) coordination and so on. In addition, node 1 assumes to be the origin node.

The xyz coordination is used to compute for rectilinear distance and euclidean distance between two nodes.

**Rectilinear distance**

Rectilinear distance is computed from xyz coordination between two nodes \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\) by using the equation (3.1).
\[ |x_1 - x_2| + |y_1 - y_2| + |z_1 - z_2| \] (3.1)

Table 3.2: Example of rectilinear distance data (meter)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>16</td>
<td>13</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>0</td>
<td>9</td>
<td>4</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>9</td>
<td>0</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>14</td>
<td>4</td>
<td>5</td>
<td>0</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>19</td>
<td>13</td>
<td>6</td>
<td>9</td>
<td>0</td>
</tr>
</tbody>
</table>

The xyz coordination from Table 3.1 is used to compute rectilinear distance as shown in Table 3.2. It is an \((N \times N)\) identical matrix where \(N\) is a number of nodes. For example, the rectilinear distance from node 1 to node 2 is calculated by \(|0 - 10| + |1 - 0| + |0 - 5| = 16\) meters.

And so on. Note that the rectilinear distance from node to node itself can automatically be assigned as zero since there is no distance.

Rectilinear distance is used for the distance of wire connection since the actual wire must be installed along the wall and/or ceiling.

**Euclidean distance**

Euclidean distance is computed from xyz coordination between two nodes \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\) by using the equation (3.2).

\[
\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}
\] (3.2)

The xyz coordination from Table 3.1 is used to compute the euclidean distance as shown in Table 3.3. It is an \((N \times N)\) identical matrix where \(N\) is a number of nodes. For example, the euclidean distance from node 1 to node 2 is calculated by \(\sqrt{(0 - 10)^2 + (1 - 0)^2 + (0 - 5)^2} = 11\) meters.
node 1 to node 3 is calculated by $\sqrt{(0 - 5)^2 + (1 - 2)^2 + (0 - 7)^2} = 18$ meters

and so on. Note that the euclidean distance from node to node itself can also automatically be assigned as zero since there is no distance.

Euclidean distance is used for the distance of wireless connection since the transmitted path is straight in all directions.

Table 3.3: Example of euclidean distance data (meter)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>11</td>
<td>8</td>
<td>9</td>
<td>11</td>
</tr>
<tr>
<td>2</td>
<td>11</td>
<td>0</td>
<td>5</td>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>9</td>
<td>3</td>
<td>0</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>8</td>
<td>5</td>
<td>7</td>
<td>0</td>
</tr>
</tbody>
</table>

Maximum distance wireless range

For making wireless connection between two nodes, the associated Euclidean distance must be within this specified distance otherwise wireless connection is not possible. For example, the maximum distance wireless range = 5 meters, therefore, node 1 and node 2 cannot use the wireless connection because the associated euclidean distance is 11 meters, which is not within distance range ($11 \text{ is more than } 5$), node 2 to node 3 can use wireless connection because the associated euclidean distance is 5 meters, which is still within the distance range ($5 \text{ is still not more than } 5$) and so on.
3.3.2 Capacity data

The capacity data category consists of the capacity of transmit using wire and wireless connection. The two capacity data have the same purpose, which is to limit the transmit signal with respective connection type.

Capacity of wire connection

Table 3.4 illustrates the capacity of wire connection data. It is an \((N \times N)\) identical matrix where \(N\) is a number of nodes. From Table 3.4, all capacities between each node of wire connection are the same, since an actual wire should transmit the same amount of limitation. For example, the capacity of wire connection between node 1 to node 2 equals 5 units, node 1 to node 3 equals 5 units and so on. In addition, there is no capacity from node to node itself.

Table 3.4: Example of the capacity of wire connection data (unit)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Capacity of wireless connection

Table 3.5 illustrates the capacity of wireless connection data. It is an \((N \times N)\) identical matrix where \(N\) is a number of nodes. From Table 3.5, all capacities between each node of wireless connection are the same, since it is the same type of wireless signal. For example, the capacity of wireless connection between node 1 to node 2 equals 5 units, node 1 to node 3 equals 5 units and so on. In addition, there is no capacity from node to node itself.
Table 3.5: Example of the capacity of wireless connection data (unit)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

3.3.3 Cost data

The cost data category is mainly used to determine the construction and operation cost of the network. Hence, it consists of the cost of locating the access point, the cost of installing physical wire connecting nodes, the wire transmission cost, and the wireless transmission cost.

Cost of locating access point

This is one of the construction costs. Whenever there is an access point located at a node, this cost must be added.

Table 3.6: Example of the cost of locating access point data (Baht)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cost (Baht)</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 3.6 illustrates the cost of locate access point data. It is a vector of \(N\) elements where \(N\) is a number of node. For example, the cost of locating an access point at node 1 equals 20 Baht, node 2 equals 20 Baht and so on. From Table 3.6, all costs of locating an access point is the same for all nodes. However, recalling that node 1 is the main router therefore the cost data can be adjusted accordingly.

Cost of installing physical wire

This is also one of the construction costs. Whenever there is a wire connection between two nodes, an actual wire is required for completing the connection.
Table 3.7: Example of the cost of installing physical wire data (Baht)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>160</td>
<td>130</td>
<td>140</td>
<td>190</td>
</tr>
<tr>
<td>2</td>
<td>160</td>
<td>0</td>
<td>90</td>
<td>4</td>
<td>130</td>
</tr>
<tr>
<td>3</td>
<td>130</td>
<td>90</td>
<td>0</td>
<td>50</td>
<td>60</td>
</tr>
<tr>
<td>4</td>
<td>140</td>
<td>40</td>
<td>50</td>
<td>0</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>190</td>
<td>130</td>
<td>60</td>
<td>90</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 3.7 illustrates the cost of installing physical wire data. It is an \((N \times N)\) identical matrix where \(N\) is a number of nodes. From Table 3.7, these data are computed from rectilinear distance multiplying by the cost per distance (in this case 10 Baht per 1 distance). For example, the cost of installing physical wire connecting node 1 to node 2 is 160 Baht, node 1 to node 3 is 130 Baht and so on.

**Wire transmission cost**

Wire transmission cost is a part of the operation costs. This data is used to calculate the transmission cost operating on wire connection. The unit is in the term of \(X\) cost per unit per distance where \(X\) is how much it costs. The full equation is the wire transmission cost times rectilinear distance times the amount of signal transmitted. For example, if the wire transmission cost = 10 Baht and 5 unit of signal is transmitted from node 1 to node 2 using wire connection, therefore, the cost is 10 x 16 x 5 = 800 Baht.

**Wireless transmission cost**

Wireless transmission cost is a part of the operation costs. This data is used to calculate the transmission cost operate on wireless connection. The unit is in the term of \(X\) cost per unit per distance where \(X\) is how much it costs. The full equation is the wireless transmission cost times euclidean distance times the amount of signal transmitted. For example, if the wireless transmission cost = 3 Baht and 4 unit signal is transmitted from node 1 to node 5 using wireless connection, therefore, the cost is 3 x 11 x 4 = 132 Baht.
3.3.4 Service data

The service data category controls the quality of the network. In this study, the service data only consists of relative demand.

Relative demand

For simplification and the algorithm perspective, the demand is standardized into relative demand attached to each node rather than the actual signal particle used in the real data communication network.

Table 3.8 illustrates the relative demand data. It is a vector of $N$ elements where $N$ is a number of nodes. For example, the relative demand of node 2 is 1, node 3 is 2 and so on. Note that node 1 has relative demand equals to 0 since node 1 is assumed to be the origin node (main router).

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Demand</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>6</td>
</tr>
</tbody>
</table>

3.4 Output data

The output data indicate how to construct the network as well as how it operates. The network output will provide a network that minimizes the total cost (the construction cost and the operation cost), while at the same time, satisfies all structural and operational requirements. The outputs are as follows:

1. The number of the access points located.
2. The location of the access points located.
3. The connection type between each node.
4. The path of transmissions to each access point as well as the associated amount of transmissions.
5. The construction cost and the operation cost, which comprise the total cost that represents the quality of the network.
CHAPTER 4

MODEL FORMULATION

This chapter explains the mathematical model formulation of the presented network. Since there are many equations in this chapter, they will be categorized into section along with some descriptions for the purpose of explanation.

4.1 Notations

The network model is formulated into Mixed Integer Programming. It defines on a directed graph $G = \{N, A\}$ where $N$ is a set of nodes and $A$ is a set of links. A pair $(i, j)$ denotes the origin $(i)$ to destination $(j)$ pairs and $(k, l)$ denotes the link between two nodes: node $(k)$ to node $(l)$ where $i, j, k, l$ is a subset of $N$. Each origin – destination pair can consist of many links $(k, l)$. For example, a route of $(0 – 4 – 3 – 5)$ is a pair of origin 0 to destination 5 $(0, 5)$ consists of nodes 0, 4, 3, and 5 with links $(0, 4), (4, 3), \text{and (3, 5)}$.

4.2 Parameters

$A_{kl}$ Rectilinear distance of link $(k, l)$
$B_{kl}$ Euclidean distance of link $(k, l)$
$C$ Maximum distance for making wireless connection possible
$D_{kl}$ Capacity of wire connection of link $(k, l)$
$E_{kl}$ Capacity of wireless connection of link $(k, l)$
$F_{kl}$ Cost of establishing wire connection of link $(k, l)$
$G_k$ Cost of locating access point at node $(k)$
$H$ Wire transmission cost
$I$ Wireless transmission cost
$J_k$ Relative demand at node $(k)$
$M$ Big number
4.3 Decision variables

\( x_k \) 1 if access point (k) is located, 0 otherwise

\( z_{kl} \) 1 if link (k, l) is using wire connection, 0 otherwise

\( w_{kl} \) 1 if link (k, l) is using wireless connection, 0 otherwise

\( y_{ijkl} \) The total amount of transmission from origin (i) to destination (j) pair (i, j) through link (k, l) regardless of the connection type

\( v_{kl} \) The total amount of transmission occurring on wireless connection of link (k, l)

4.4 Mathematical model equation

4.4.1 Objective function

\[
\text{Minimize } Z = \sum_{k} \sum_{l} (F_{kl} \cdot z_{kl}) + \sum_{k} \left( G_k \cdot x_k \right) + \sum_{i} \sum_{j} \sum_{k} \sum_{l} \left( H \cdot A_{kl} \cdot y_{ijkl} \right) + \sum_{k} \sum_{l} \left( H \cdot \left( \frac{1}{H} \cdot B_{kl} \right) - A_{kl} \right) \cdot v_{kl} \quad (4.1)
\]

The objective function (Equation 4.1) is to minimize the total cost, which consists of construction cost and operation cost. The first term is the cost of installing physical wire for all wire connections. The second term is the cost of locating access point. The third term is the transmission cost operating on the wire connection. Finally, the forth term is the transmission cost operating on the wireless connection.
4.4.2 Constraints

*Flow conservation constraint*

\[
\sum_l y_{ijkl} - \sum_l y_{ijkl} = \begin{cases} 
J_j & \forall i, j, k \in N \\
0 & \forall i, j, k \in N 
\end{cases}
\]  

Equation (4.2) ensures the flow conservation of each origin – destination pair at every associate node \((l)\). For origin \((i)\) to destination \((j)\), the flow in minus flow out of every node \((l)\) will be: relative demand of destination \((j)\) if that node is an origin node \((i)\) itself, negative relative demand of destination \((j)\) if that node is a destination \((j)\) itself, and 0 if that node is neither origin nor destination (intermediate node).

*Wireless portion constraints*

\[
v_{kl} \geq \sum_l \sum_j y_{ijkl} - (M * z_{kl}) \quad \forall k, l \in N
\]  

\[
v_{kl} \leq \sum_l \sum_j y_{ijkl} \quad \forall k, l \in N
\]  

Equation (4.3) and (4.4) represent the transmission amount of wire and wireless connection. Since \(y_{ijkl}\) is the total amount of transmission regardless of the connection type whereas \(v_{kl}\) is the total amount of wireless transmission, therefore \(y_{ijkl}\) can be represented as the total amount of wire transmission. The wireless portion is then compensated by \(v_{kl}\).

*One connection type constraint*

\[
w_{kl} + z_{kl} \leq 1 \quad \forall k, l \in N
\]  

(4.5)
Equation (4.5) allows only one type of connection per every link \((k, l)\). In other words, this ensures the single type of transmission on every link \((k, l)\).

Capacity constraint

\[
\sum_j \sum_l y_{ijkl} \leq (D_{kl} \times z_{kl}) + (E_{kl} \times w_{kl}) \quad \forall k, l \in N \tag{4.6}
\]

Equation (4.6) will not allow the amount of transmission for connected link to exceed the capacity for its respective connection type.

Access point requirement constraints

\[
z_{kl} \leq x_k \quad \forall k, l \in N \tag{4.7}
\]

\[
z_{kl} \leq x_l \quad \forall k, l \in N \tag{4.8}
\]

\[
w_{kl} \leq x_k \quad \forall k, l \in N \tag{4.9}
\]

\[
w_{kl} \leq x_l \quad \forall k, l \in N \tag{4.10}
\]

The wire and wireless connection of link \((k, l)\) requires an access point at both node \((k)\) and \((l)\). Equation (4.7) and (4.8) are the access point requirement of wire connection whereas equation (4.9) and (4.10) are the access point requirement of wireless connection.

Wireless usability constraints

\[
M \times v_{kl} \geq w_{kl} \quad \forall k, l \in N \tag{4.11}
\]

\[
M \times w_{kl} \geq v_{kl} \quad \forall k, l \in N \tag{4.12}
\]
The wireless transmission must be transmitted on the wireless connection – and vice versa – the wireless connection will be established only if there is a need for wireless transmission, as expressed by equation (4.11) and (4.12) respectively.

**Wireless connection range constraint**

\[
B_{kl} \cdot w_{kl} \leq C \quad \forall k, l \in N
\]  

(4.13)

In order to establish a wireless connection of link \((k, l)\), the euclidean distance of that link must be within the pre-specified range. This limitation is expressed by equation (4.13).

**Binary and non-negativity constraints**

\[
x_{kl}, z_{kl}, w_{kl} \in \{0, 1\} \quad \forall k, l \in N
\]  

(4.14)

\[
y_{ijkl}, v_{kl} \geq 0 \quad \forall i, j, k, l \in N
\]  

(4.15)

The binary and non-negativity constraints for each decision variable are expressed by equation (4.14) and (4.15).
CHAPTER 5
METHODOLOGY

5.1 Branch and cut algorithm

The model presented in this study is tested with a branch and cut algorithm using CPLEX software implemented on Concert technology with C++. The branch and cut algorithm is guaranteed to yield an optimal result. The maximum computation time is set at 5,000 seconds. Hence, there are three scenarios on this testing methodology:

1. The computation time does not exceed 5,000 seconds. In this case, the optimal solution is found.
2. CPLEX cannot complete the computation time within 5,000 seconds. Therefore, it will return the best feasible solution found.
3. If the branch and cut algorithm heavily consumes memory, the computation runs out of the memory. There will be no solution.

5.2 Proposed heuristic algorithm

The basic principle of heuristics is to find the acceptable solution in the large amount of possible solution as fast as possible. It also needs an evaluation process to know whether the found solution is acceptable or not. In summary, the main idea of the heuristics is to generate and evaluate the solution.

The proposed LS algorithm is divided into three stages. Each consecutive stage aims for progressive enhancement from the previous stage results with the last stage as the best of all stages. All three algorithms are implemented using Julia programming language with version (0.3.7). These three stages are as follows:

1. Basic iterated LS algorithm (LS1).
2. LS algorithm with Improvise Search idea from HS algorithm (LS2).
3. **LS algorithm** that derives an implementation from the objective function to help the solution generating process more efficiently (LS3).

Each stage is different in the idea of the solution generating process whereas the evaluation process is the same for all stages. Therefore, the understanding of the solution and evaluation process is the first priority.

The implemented code of the proposed algorithm can be seen in Appendix A. The code in Appendix A is the final structure of the algorithm. Only changing the parameters or commenting some statements can result in all 3 stages of LS. More details are explained inside the code.

5.2.1 Solution representation

The proposed heuristic yields 3 solutions according to the output data. They are the connection type matrix (main solution), the AP location vector, and the transmission matrix.

**Main solution matrix**

The main solution that the heuristics needed to generate is in the form of identical matrix \((N \times N)\) where \(N\) is the number of nodes. Each cell in the solution matrix represents the connection type. The connection type used numerical data to represent different connection types as follows:

- **a.** \(-1\) – (negative one) indicates that the two nodes cannot be connected to each other. This is only used for the diagonal of the matrix.
- **b.** \(0\) – indicates that the two nodes do not connect each other.
- **c.** \(1\) – indicates that the two nodes connect each other using the wire connection.
- **d.** \(3\) – indicates that the two nodes connect each other using the wireless connection.

Table 5.1 illustrates the solution matrix representation. For example, the connection between node 1 to node 2 uses the wire connection, node 3 to node 5 uses the wireless connection, node 4 to node 2 is not connected to each other. In addition, the connection between node to node itself cannot be made, so negative number is used to indicate that.
Table 5.1: Example of solution matrix

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>-1</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>-1</td>
</tr>
</tbody>
</table>

**AP location vector**

AP location vector indicates which node will locate the access point. In this data type, numerical 1 represents a located access point whereas 0 represents the opposite. It is a vector of $N$ elements where $N$ is a number of nodes.

Table 5.2: Example of AP location vector

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>AP</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 5.2 illustrates the AP location vector output. For example, AP needs to be located at node 1, node 3, and node 4 whereas node 2 and node 5 need not.

**Transmission matrix**

Another output is the transmission matrix. It is a $(N \times N)$ matrix where $N$ is a number of nodes. This matrix is not an identical matrix unlike the others because it is needed to indicate a direction of transmission.

Table 5.3 illustrates the transmission matrix output. It is translated into a path with a direction of transmission and corresponding with the connection type matrix to obtain the transmission type. For example, the translated output can be:

- Node 1 to node 2 with amount of transmission equal to 1 unit
- Node 1 to node 3 with amount of transmission equal to 5 units
- Node 1 to node 4 with amount of transmission equal to 1 unit
- Node 1 to node 5 with amount of transmission equal to 3 units
- Node 3 to node 5 with amount of transmission equal to 3 units
- Node 4 to node 2 with amount of transmission equal to 3 units

Table 5.3: Example of transmission matrix (unit)

<table>
<thead>
<tr>
<th>Node</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

5.2.2 Solution evaluating process

The evaluation process employs the shortest path Dijkstra algorithm to iteratively solve the shortest distance and augmenting the flow from origin to each destination. The followings are the input for starting the evaluation algorithm:

1. Generating connection type (main solution)
2. Associated instance data

The algorithm will try to transmit the supply for the demand while keeping the respect to the associated capacity. If the algorithm cannot complete the requirement, the solution is marked as an infeasible solution and large penalty cost will be added to the total cost. Otherwise, the algorithm will return the followings output:

1. The total cost of construction and transmission cost.
2. Optimize connection type where unused connection will be cut off.
3. AP location according to the optimized connection type.
4. Transmission path and associate amount.

5.3 LS1 algorithm

The algorithm LS1 is a basic LS with iteration stopping criteria. The algorithm keeps track only one best solution. After the new candidate solution is generated and then evaluated, if the new one is better, it will replace the stored solution with the new solution. Then, keep doing it until the stopping criteria is reached. Figure 5.1 illustrates the flowchart of LS1 algorithm.
5.3.1 LS1 solution generating process

Recall that in order to generate a solution, the algorithm will go through each cell in the matrix solution and then assign the value of connection type. In LS1, each cell has two behaviors of assigning the value by the probability of the behavior.

Figure 5.2 illustrates 2 possible actions to assign the value to each cell: F1 and F2. Suppose the algorithm currently wants to assign the value at the focused cell (indicated by red square), F1 will use the same value of the exact same cell position of the stored solution. For F2, randomly pick 1 or 3 (wire or wireless) and assign that value immediately. The probability of F1 and F2 is 0.5 and 0.5 respectively.
5.4 LS2 algorithm

The algorithm LS2 is modified based on LS1. While LS1 only stores one candidate solution, LS2 utilizes a memory and solution generating process concept from HS. Therefore, instead of initializing dummy best candidate solution waiting to be removed, LS2 uses the same procedure as HS, which initializes a set of solutions and stores them inside HM. This process is described as initialize HM and the purpose is that the algorithm will have an initial memory to start with. Figure 5.3 illustrates the LS2 algorithm flowchart. Since LS2 is influenced by HS, the next section will introduce the HS algorithm.

5.4.1 HS algorithm

HS algorithm stores a number of candidate solutions in a memory called Harmony Memory (HM). The process of generating a solution is called Improvise Search. The basic process of HS is, first, generating a set of candidate solutions and storing them inside HM. This will give the initial solution for the HS algorithm to
work with. Then, HS algorithm will generate the solution using Improvise Search and evaluate that generated solution. If the evaluated solution is better than the worst solution inside HM, replace the worst one with the new one. Keep generating and evaluating until the stopping criteria is reached. Thus, the best solution inside HM is reported as the final result. There are 4 main parameters used inside the algorithm:

- Harmony Memory Size (HMS): A number of candidate solutions stored inside HM.
- Iteration Stopping Criteria (ITER): A number of iteration to perform the algorithm.
- Harmony Memory Consideration Rate (HMCR) and Pitch Adjustment Rate (PAR): These two parameters will be used to determine the behavior of Improvise Search.

![Figure 5.3: LS2 algorithm flowchart](image)

Figure 5.3: LS2 algorithm flowchart
There are 3 behaviors of Improvise Search. Note that only one behavior is used per one value. Therefore, each cell in the solution matrix will utilize only one behavior to assign the connection type value. They are as follows:

- Memory Consideration (MC): with probability of \([HMCR \times (1 - PAR)]\)
- Pitch Adjustment (PA): with probability of \([HMCR \times PAR]\)
- Random Selection (RS): with probability of \([1 - HMCR]\)

Figure 5.4 illustrates 3 behavior actions of Improvise Search. In Figure 5.4, there are 3 candidate solutions stored inside HM. The bottom rectangle represents the new solution that is in the solution generating process.

Suppose that the algorithm wants to assign the value to the focused cell (indicated by red square), MC will gather all values of the exact same cell position for each stored candidate solution (indicated by green square), randomly choose one and assign that value. For PA, the process is the same as MC, but before assigning the value, there is an adjustment to the current value. Lastly, RS just randomly chooses either 1 or 3 (wire or wireless) and assign the value immediately.
5.4.2 LS2 solution generating process

If HS is utilized in full implementation, one obvious problem is that there are many possible combinations of parameter setting values. For HMCR and PAR, the probability can be ranged from [0.0, 1.0]. Finding the best possible parameter setting for each instance is not an efficient methodology. In addition to that, the performance can be bias from how the Improvise Search behavior (MC, PA, and RS) is implemented. Afkhami et al. (2013) implemented Harmony Improvisation on a binary problem.

It should be noted that, MC of HS and F1 of LS1 use the same function but the difference is the number of candidate solution being stored and how to choose the value. Also RS of HS and F2 of LS1 are entirely the same function. Therefore, LS2 will utilize only MC and RS with the parameter setting of HMCR and PAR equal to 0.5 and 0.0, respectively.

5.5 LS3 algorithm

LS3 uses the same algorithm as LS2. However, LS3 contains an implementation that is derived from the objective function in order to aid the solution generating process more effectively. In LS3, the algorithm will try to guide the solution generating process in which what connection type should be used in a particular spot.

5.5.1 LS3 solution generating process

Before assigning the value of connection type, the algorithm will calculate the cost per unit of both wire and wireless connection types. The corresponding value to be assigned is determined which cost per unit of connection type is lower due to minimizing the total cost of the objective function. If the wire cost per unit is lower than the wireless cost per unit, that cell value is then assigned to be the wire connection type. Otherwise is should be assigned to be the wireless connection.
Equation (5.1) and (5.2) are the equations for the wireless cost per unit and wire cost per unit, respectively. Both equations use the parameters illustrated in Chapter 3.
CHAPTER 6
DESIGN OF EXPERIMENTS

The experiments conducted on 9 classes of random test instances. Each class is different in terms of problem size, represented by the number of nodes ranging from 20, 30, 40, 50, 60, 70, 80, 90, and 100. All classes consist of 40 different instances with two different structures (A and B). First 20 instances use structure A whereas the other half uses structure B. The total number of instances is 360. Table 6.1 illustrates the random range of both data structures.

Table 6.1: Data structure A and B

<table>
<thead>
<tr>
<th>Data</th>
<th>Structure A</th>
<th>Structure B</th>
</tr>
</thead>
<tbody>
<tr>
<td>XYZ coordination</td>
<td>[0, 10]</td>
<td>[0, 100]</td>
</tr>
<tr>
<td>Demand</td>
<td>[0, 10]</td>
<td>[0, 10]</td>
</tr>
<tr>
<td>Cost of locating access point</td>
<td>20</td>
<td>200</td>
</tr>
<tr>
<td>Wire capacity</td>
<td>Adjust according to the demand data.</td>
<td>Adjust according to the demand data.</td>
</tr>
<tr>
<td>Wireless capacity</td>
<td>Adjust according to the demand data.</td>
<td>Adjust according to the demand data.</td>
</tr>
<tr>
<td>Wire transmission cost</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Wireless transmission cost</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Max wireless distance</td>
<td>5</td>
<td>60</td>
</tr>
<tr>
<td>Rectilinear distance</td>
<td>Compute from XYZ coordination.</td>
<td>Compute from XYZ coordination.</td>
</tr>
<tr>
<td>Euclidean distance</td>
<td>Compute from XYZ coordination.</td>
<td>Compute from XYZ coordination.</td>
</tr>
<tr>
<td>Cost of installing physical wire</td>
<td>Rectilinear distance multiplies by 10.</td>
<td>Rectilinear distance multiplies by 10.</td>
</tr>
</tbody>
</table>

Each instance data is encapsulated within its own text file (.txt) contains: XYZ coordination, demand, cost of locating access point, wire capacity, wireless capacity, wire transmission cost, wireless transmission cost, and maximum wireless distance. Note that rectilinear distance, euclidean distance, and cost of installing physical wire will be created within the runtime. The reason behind this is that those three data require other data to compute as discussed in Chapter 3. Therefore, it would be better
to create them within the runtime so that the data in the text file can be changed easily. Example of the text file of structure A and B with node size equal to 20 can be seen in Appendix B.

There are two scenarios in order to generate each random data instance:

1. Each XYZ coordination must not be the same position. It does not make sense to have the same exact position of nodes.

2. Both wire and wireless capacities must be greater than the total sum of demand. If both capacities are less than the total sum of demand, the network cannot be solved because the demand will not be satisfied. If capacity * (N – 1) is less than the total sum of demand, increased the capacity by 1 until it is greater, then assign the current capacity value to both wire and wireless capacity. Note that in this study, the starting value of capacity equals to 5.

The above scenarios will apply to each instance separately. It is important to understand that each instance is an entirely different entity and there is no relation between others. All test instances are run on Intel Core i5 2.5 GHz CPU with 8.00 GB RAM. All 3 LS algorithms stopping criteria is based on a number of iterations. Therefore, the iteration stopping criteria is set at 5,000 iterations.
CHAPTER 7
RESULTS AND DISCUSSIONS

Both CPLEX and proposed LS algorithm network results are obtained in the form of text file (.txt). Each result is tested by not allowing to violate the constraint regulation. Therefore, the results can be discussed on the performance comparison. Example of the network result in text file can be seen in Appendix B.

The results in this section focus mainly on the comparison of the obtained objective function from each algorithm, which represents in form of gap. In addition, the runtime will also be discussed. Each algorithm results in 2 tables, which are the objective value and the runtime. Please refer to Appendix C for the full results table.

The comparison is divided into two types: comparison to CPLEX and comparison to LS3. Comparison to CPLEX is used to discuss on node size from 20 up to 60 whereas the latter is on 70 up to 100. This is because as node size reaches 70, CPLEX is unable to solve some of the instances. As node size equals 100, CPLEX cannot solve any instances at all.

7.1 Comparison to CPLEX

Only node size of 20, 30, 40, 50, and 60 can be compared with CPLEX due to the memory issue of branch and cut algorithm. This section focuses on gap comparison and runtime comparison. The gap comparison is the percentage difference of the objective value in each LS algorithm to CPLEX. The calculation is needed in order to see the difference rather than comparing the objective value directly. For the runtime comparison, the runtime itself is being compared.

7.1.1 Gap comparison

\[
\left( \frac{LSx - CPLEX}{CPLEX} \right) \times 100
\] (7.1)
Figure 7.1: Percentage gap of LSx as compared to CPLEX
Equation (7.1) is used to calculate the gap or the percentage difference of each LS to CPLEX. \(LS_x\) illustrates the objective value of LS1, LS2, and LS3.

Figure 7.1 illustrates the gap as compared to CPLEX of the node size from 20 up to 60. The y-axis represents the percentage gap whereas the x-axis represents the instances. Recall that each node size contains 40 different instances as well as the first 20 instances are the data structure A and another 20 instances are the data structure B. Therefore, in the x-axis, instance 1 to instance 40 belongs to node size of 20, instance 41 to instance 80 belongs to node size of 30, and so on. LS1, LS2, and LS3 use color blue, red, and green respectively. As seen from the graph, at node size of 20, all 3 LS algorithms perform quite similar to each other. However, as the node size increases, LS1 and LS2 fluctuation is increased as well as the percentage gap increases whereas LS3 fluctuation tends to decrease.

Figure 7.1 shows the gap of LS1, LS2, and LS3 algorithm of each instance of node size 20, 30, 40, 50, and 60 as compared to CPLEX. LS3 is shown to clearly perform better than LS1 and LS2.

![Figure 7.2: Average percentage gap of LSx as compared to CPLEX](image)

Figure 7.2 illustrates the average gap of each node size ranging from 20 to 60. The y-axis represents the percent gap as compared to CPLEX whereas the x-axis represents the node size. Performances of LS1, LS2, and LS3 are indicated by blue, red, and green color, respectively. Figure 7.3 illustrates the standard deviation gap of
each node size ranging from 20 to 60. The y-axis represents the standard deviation gap whereas the x-axis represents the node size. LS1, LS2, and LS3 are indicated by blue, red, and green color, respectively.

From Figure 7.2 and 7.3, as the node size increases, LS1 and LS2 lose to LS3. Both average and standard deviation gap of LS3 are not much fluctuating as the node size increases.

![Figure 7.3: Standard deviation gap of LSx](image)

7.1.2 Runtime comparison

The runtime comparison compares the actual CPU process time of CPLEX, LS1, LS2, and LS3. Again, only node size of 20 up to 60 is used as illustrated in Figure 7.4. The y-axis represents the runtime in second whereas the x-axis is the instances. Performances of CPLEX, LS1, LS2, and LS3 are indicated by blue, red, green, and purple color respectively.

LS1, LS2, and LS3 will increase in their runtime as the node size increases. It is because there is more computation required as the node size increases. CPLEX also becomes ineffective due to the memory consumption of the branch and cut algorithm.

Although LS1, LS2, and LS3 use roughly similar runtime, LS3 results in a better objective value as illustrated by the gap comparison from the previous section.
Figure 7.4: Runtime Comparison
Figure 7.5 illustrates the average time of each node size ranging from 20 to 60. The y-axis represents the time in second whereas the x-axis represents the node size. Performances of CPLEX, LS1, LS2, and LS3 are indicated by blue, red, green, and purple color respectively.

![Figure 7.5: Average of runtime comparison](image)

Figure 7.6 illustrates the standard deviation time of each node size ranging from 20 to 60. The y-axis represents the standard deviation time in second whereas

![Figure 7.6: Standard deviation of runtime comparison](image)
the x-axis represents the node size. Performances of CPLEX, LS1, LS2, and LS3 are indicated by blue, red, green, and purple color respectively.

CPLEX is not suitable for big problem size as its average runtime and standard deviation of runtime is too large, while LS1, LS2, and LS3 show to perform reasonably well when the node size is large.

7.2 Comparison to LS3

This section focuses on the gap comparison of LS1, LS2, and LS3 to LS3. It is because starting at node size of 70, CPLEX cannot solve some instances. Since the previous section clearly indicates that LS3 performance is acceptable, therefore, this section will used LS3 as a base performance.

Equation (7.2) is used to calculate the percentage different gap of each LS to LS3. LSx is used to illustrate the objective value of LS1, LS2, and LS3.

$$\left( \frac{LSx - LS3}{LS3} \right) \times 100$$  \hspace{1cm} (7.2)

Figure 7.7 illustrates the graph of the percentage gap comparison of LS1, LS2, and LS3 to LS3. Note that as LS3 is compared against LS3 itself, therefore, the green line will always be at the bottom of the graph. To interpret this graph, if red and blue lines, which indicate LS1 and LS2 respectively, touch the green line, it means that the objective value is the same as LS3. Unsurprisingly, both LS1 and LS2 performance cannot be compared to LS3 performance as indicated by the fact that both lines are always above the line of LS3.
Figure 7.7: Percentage gap of LSx as compared to LS3
CHAPTER 8
CONCLUSION AND RECOMMENDATION

This study developed an algorithm to solve the unique underlying network design problem. The algorithm is based on Local Search heuristics. The network in this study contains its own characteristics and behaviors. Having gathered all requirements, the network was then formulated into Mixed Integer Programming model. This mathematical model was tested with a branch and cut algorithm using CPLEX concert technology with C++. Since branch and cut algorithm can guarantee the optimal solution, therefore, the proposed algorithm can be analyzed effectively.

There are three stages of the proposed algorithm: LS1, LS2, and LS3. LS1 is the basic iterated Local Search. LS2 utilizes Harmony Memory and Improvise Search of Harmony Search in the solution generating process. Finally, LS3 uses the equation from the objective function to guide the proper connection type, also in the solution generating process. All three algorithms were implemented using Julia programming language version 0.3.7.

The experiment was conducted on total of 360 data instances. Each 40 instances are different in the number of node size. There are node size of 20, 30, 40, 50, 60, 70, 80, 90, and 100. Also, there are 2 patterns of data structure (A and B), dividing for each 20 instances of each node size. Each algorithm was then used to solve all 360 instances.

The results from analysis and evaluation stated that LS3 yields the objective value closest to the value from the branch and cut algorithm. For the performance of obtaining the objective value, the performance of LS1, LS2, and LS3 are quite similar at the node size of 20. At the node size of 30, the performance of LS3 is apparently better than LS1 and LS2, and continuing to be better as the node size increases. For the algorithm runtime performance, the runtime of LS1, LS2, and LS3 are approximately the same with similar node size.

There is one interesting aspect of implementing the proposed algorithm. In this study, the shortest path Dijkstra algorithm is used as a main algorithm in the
evaluation process. Dijkstra algorithm is a time consuming algorithm. Therefore, the proposed algorithm needs to run Dijkstra algorithm as minimum as possible. Thus, the trade-off should be taken into consideration as follows:

- If the proposed algorithm find more feasible solution, which leads to utilize more Dijkstra algorithm. The advantage is that there are more chances of obtaining better solution whereas the disadvantage is more time consuming.
- If the proposed algorithm finds less feasible solution, which leads to utilize less Dijkstra algorithm. The advantage is less time consuming whereas the disadvantage is less chance of obtaining a better solution.

The evaluation process in the proposed LS algorithm does not have the permission of changing the generated solution and it only receives the generated solution and tries to evaluate the given solution. It would be interesting if the evaluation process can obtain the permission to change the generated solution. This could make the feasible solution to become a better solution and/or change the infeasible solution into a feasible solution. Although the solution generating process and the solution evaluating process already work together, both processes should work simultaneously not separately.
REFERENCES


APPENDIX A

PROPOSED ALGORITHM CODE

Appendix A shows the proposed algorithm code. The code is rearranged into the final structure, which can achieve LS1, LS2, and LS3 algorithm by changing the parameters and/or commenting/uncommenting corresponding statements. The algorithm code is implemented using Julia programming language version 0.3.7. There are total of 10 files for the proposed algorithm as follows:

1. bestSolution.jl – uses for keeping track of the best solution.
2. config.jl – uses for configuring parameters and initialize constants.
3. dijkstra.jl – the implementation of Dijkstra algorithm.
4. evaluate.jl – uses for the solution evaluating process.
5. generate.jl – uses for the solution generating process.
6. input.jl – uses for reading data from the text file as well as creating some data on runtime.
7. main.jl – the main control flow of the algorithm.
8. memory.jl – uses for keeping track of the memory that uses to store candidate solution.
9. solution.jl – uses for storing the information of the new solution.
10. utils.jl – utility functions.
using Config
export Best

type Best
    time::Float64
    total_obj::Int
    fixed_obj::Int
    trans_obj::Int
    trans_memory::Matrix{Int}
    ap_location::Vector{Int}
    connect_type::Matrix{Int}

    function Best(nsize::Int)
        time = 0.0
        total_obj = M
        fixed_obj = 0
        trans_obj = 0
        trans_memory = zeros(Int, (nsize, nsize))
        ap_location = zeros(Int, nsize)
        connect_type = zeros(Int, (nsize, nsize))

        new(time,
            total_obj,
            fixed_obj,
            trans_obj,
            trans_memory,
            ap_location,
            connect_type)
    end
end
module Config

export SIZE, VARIANT, HMS, HMCR, PAR, ITER, M

# node size will run in loop over this array
const SIZE = [20, 30, 40, 50, 60, 70, 80, 90, 100]

# number of instances per node size
const VARIANT = 40

# If set HMS = 1, then the algorithm is LS1
const HMS = 50
const HMCR = 0.5
const PAR = 0.0
const ITER = 5000
const M = 10^6

end
module Dijkstra

using Config

export dijkstra, find_path

function min_idx(A::Vector{Int}, B::Vector{Int})
    min_num = 10^8
    min_index = 0
    for idx = 1:length(A)
        if B[idx] == 0 && A[idx] < min_num
            min_num = A[idx]
            min_index = idx
        end
    end
    return min_index
end

function find_path(pred::Vector{Int}, dest::Int)
    paths = Int[pred[dest], dest]
    node = pred[dest]
    while node > 1
        unshift!(paths, pred[node])
        node = pred[node]
    end
    return paths
end

function dijkstra(graph::Matrix{Int})
    nsize = size(graph, 2)
    visited = zeros(Int, nsize)
    dist = fill(M, nsize)
    pred = ones(visited)
    pred[1] = -1

dist[1] = 0
while 0 in visited
    i = min_idx(dist, visited)
    visited[i] = 1
    for j = 1:nsize
        if graph[i, j] != 0 && graph[i, j] != M
            if dist[j] > dist[i] + graph[i, j]
                dist[j] = dist[i] + graph[i, j]
                pred[j] = i
            end
        end
    end
end
return dist, pred
using Config
using Dijkstra

function create_distance_capacity!(data::InputData, solution::PotentialSolution, nsize::Int)
    for col=1:nsize, row=col+1:nsize
        if solution.connect_type[row,col] == 1
            dist = *(data.recti[row,col], data.lan_transfer_cost)
            cap = data.lan_cap[row,col]
        elseif solution.connect_type[row,col] == 3
            dist = *(data.eucli[row,col], data.wlan_transfer_cost)
            cap = data.wlan_cap[row,col]
        else
            dist = M
            cap = 0
        end
        solution.distance[row,col] = dist
        solution.capacity[row,col] = cap
        solution.distance[col,row] = dist
        solution.capacity[col,row] = cap
    end
    nothing
end

function is_enough_capacity(data::InputData, solution::PotentialSolution, nsize::Int)
    cap_out_origin = sum(solution.capacity[:,1])
    total_demand = sum(data.work_demand)
    if cap_out_origin < total_demand
        return false
    end
    if any(sum(solution.capacity, 1) .< data.work_demand)
        return false
    end
    return true
end
function index_min(A::Vector{Int}, B::Vector{Int})
    min_num = 10^8
    min_index = 10^8
    for idx=1:length(A)
        if B[idx] != 0 && A[idx] < min_num
            min_num = A[idx]
            min_index = idx
        end
    end
    return min_index
end

function path_min_cap(solution::PotentialSolution, path::Vector{Int})
    cap = 10^8
    for i=2:length(path)
        target_cap = solution.capacity[path[i-1], path[i]]
        if target_cap < cap
            cap = target_cap
        end
    end
    return cap
end

function disconnect!(solution::PotentialSolution, nsize::Int)
    for col=1:nsize, row=1:nsize
            solution.connect_type[row,col] = 0
            solution.capacity[row,col] = 0
            solution.distance[row,col] = M
        end
    end
    nothing
end

function transfer!(data::InputData, solution::PotentialSolution, nsize::Int)
    while true
        is_still_trans = false
        dist, pred = dijkstra(solution.distance)
while minimum(dist) != M
    dest = index_min(dist, data.work_demand)
    if dest == 10^8
        break
    end
    path = find_path(pred, dest)
    path_cost = dist[dest]
    path_cap = path_min_cap(solution, path)
    trans = data.work_demand[dest] < path_cap ? data.work_demand[dest] : path_cap

    if trans > 0
        for (idx, node) in enumerate(path)
            if idx > 1
                pred_node = path[idx - 1]
                solution.capacity[pred_node, node] -= trans
                solution.capacity[node, pred_node] -= trans
                solution.trans_memory[pred_node, node] += trans
                if solution.capacity[pred_node, node] == 0
                    solution.distance[pred_node, node] = M
                    solution.distance[node, pred_node] = M
            end
        end
        solution.trans_obj += trans * path_cost
        data.work_demand[dest] -= trans
        dist[dest] = M
        is_still_trans = true
    else
        is_still_trans = true
        break
    end
    if !is_still_trans
        break
    end
end

if any(demand -> demand > 0, data.work_demand)
    solution.trans_obj = M
else
solution.is_feasible = true
disconnect!(solution, nsize)
end
nothing
end

function fixed!(data::InputData, solution::PotentialSolution, nsize::Int)
    for col=1:nsize, row=1:nsize
        if solution.trans_memory[row,col] > 0
            solution.ap_location[row] = 1
            solution.ap_location[col] = 1
        end
    end
    for col=1:nsize, row=1:nsize
        if solution.connect_type[row,col] == 1 && solution.trans_memory[row,col] > 0
            solution.fixed_obj += data.lan_cost[row,col]
        end
    end
    solution.fixed_obj += sum(data.ap_cost .* solution.ap_location)
end
nothing

function evaluate!(data::InputData, solution::PotentialSolution, nsize::Int)
    data.work_demand = map(i -> i, data.demand)
    create_distance_capacity!(data, solution, nsize)
    if is_enough_capacity(data, solution, nsize)
        transfer!(data, solution, nsize)
        if solution.is_feasible
            fixed!(data, solution, nsize)
        else
            solution.fixed_obj = M
        end
    else
        solution.trans_obj = M
        solution.fixed_obj = M
    end
    solution.total_obj = solution.trans_obj + solution.fixed_obj
end
nothing
using Config

function cost_per_unit(data::InputData, row::Int, col::Int)
    wlan = data.wlan_transfer_cost * data.eucli[row,col]
    lan = (data.lan_transfer_cost * data.recti[row,col]) + (data.lan_cost[row,col] / data.lan_cap[row,col])
    wlan > lan ? 1 : 3
end

function random_connect_type!(data::InputData, solution::PotentialSolution, nsize::Int)
    for col=1:nsize, row=col+1:nsize
        if data.eucli[row,col] == M
            connect = 1
        else
            connect = rand() <= 0.5 ? 1 : 3
        end
        solution.connect_type[row,col] = connect
        solution.connect_type[col,row] = connect
    end
    nothing
end

function memory_consideration(memory::HarmonyMemory, row::Int, col::Int)
    memory.connect_type[rand(1:end)][row,col]
end

function pitch_adjustment(memory::HarmonyMemory, row::Int, col::Int)
    mc = memory.connect_type[rand(1:end)][row,col]
    if mc == 0
        return rand() <= 0.5 ? 1 : 3
    elseif mc == 1
        return rand() <= 0.5 ? 0 : 3
    else
        return rand() <= 0.5 ? 1 : 0
    end
function random_selection()
    rand() <= 0.5 ? 1 : 3
end

function improvise_search!(data::InputData, solution::PotentialSolution, memory::HarmonyMemory, nsize::Int, HMCR::Float64, PAR::Float64)
    for col=1:nsize, row=col+1:nsize
        if rand() < HMCR
            if rand() < PAR
                connect = pitch_adjustment(memory, row, col)
            else
                connect = memory_consideration(memory, row, col)
            end
        else
            connect = random_selection()
        end
        if connect == 3 && data.eucli[row,col] == M
            connect = 1
        end
        # Comment the following if statement to disable LS3 equation.
        if connect == 1 && data.eucli[row,col] != M
            connect = cost_per_unit(data, row, col)
        end
        solution.connect_type[row,col] = connect
        solution.connect_type[col,row] = connect
    end
    nothing
end
Appendix A-6

Proposed Algorithm Code – input.jl

module Input

using Config

export InputData

type InputData
    xyz::Matrix{Int}
    recti::Matrix{Int}
    eucli::Matrix{Int}
    demand::Vector{Int}
    ap_cost::Vector{Int}
    lan_cost::Matrix{Int}
    lan_cap::Matrix{Int}
    lan_transfer_cost::Int
    wlan_transfer_cost::Int
    wlan_cap::Matrix{Int}
    max_wlan_distance::Int
    work_demand::Vector{Int}

function InputData(nsize::Int, variant::Int)
    file = open(string("./data/n", nsize, "/instance_", nsize, "-", variant, ".txt"))
    xyz = read_to_matrix(file, nsize, 3)
    recti = rectilinearization(xyz, nsize)
    eucli = euclideanization(xyz, nsize)
    demand = read_to_array(file)
    ap_cost = read_to_array(file)
    lan_cost = recti * 10
    lan_cap = read_to_matrix(file, nsize)
    wlan_cap = read_to_matrix(file, nsize)
    lan_transfer_cost = read_to_item(file)
    wlan_transfer_cost = read_to_item(file)
    max_wlan_distance = read_to_item(file)
close(file)

work_demand = zeros(demand)

map!((x) -> x > max_wlan_distance ? M : x, eucli)

new(xyz, recti, eucli, demand, ap_cost, lan_cost, lan_cap, lan_transfer_cost, wlan_transfer_cost, wlan_cap, max_wlan_distance, work_demand)

function read_to_matrix(file, nsize, size_horizontal=nsize)
    matrix = zeros(Int, (nsize, size_horizontal))
    readline(file)
    for i=1:nsize
        for (index, item) in enumerate(split(readline(file)))
            matrix[i,index] = int(item)
        end
    end
    return matrix
end

function read_to_array(file)
    array = Int[]
    readline(file)
    for item in split(readline(file))
        push!(array, int(item))
    end
    return array
end
function read_to_item(file)
    readline(file)
    return int(readline(file))
end

function rectilinearization(xyz::Matrix{Int}, nsize::Int)
    recti = zeros(Int, (nsize, nsize))
    for i=1:nsize, j=1:nsize
        x = abs(xyz[i,1] - xyz[j,1])
        y = abs(xyz[i,2] - xyz[j,2])
        z = abs(xyz[i,3] - xyz[j,3])
        recti[i,j] = +(x, y, z)
    end
    return recti
end

function euclideanization(xyz::Matrix{Int}, nsize::Int)
    eucli = zeros(Int, (nsize, nsize))
    for i=1:nsize, j=1:nsize
        x = (xyz[i,1] - xyz[j,1])^2
        y = (xyz[i,2] - xyz[j,2])^2
        z = (xyz[i,3] - xyz[j,3])^2
        eucli[i,j] = floor(sqrt(+(x, y, z)))
    end
    return eucli
end

end
using BestSolution
using Config
using Input
using Memory
using Solution
include("evaluate.jl")
include("generate.jl")
include("utils.jl")
using ProgressMeter

function main()
    for nsize in SIZE
        overview_file = string("../output/n", nsize, "/overview.txt")
        of = open(overview_file, "w")
        solution = PotentialSolution(nsize)

        for variant = 1:VARIANT
            best = Best(nsize)
            input = InputData(nsize, variant)
            memory = HarmonyMemory()

            println(nsize, "--", variant)
            progress = Progress(ITER+HMS, 1, "Computing...", 25)
            starttime = time()

            for _ = 1:HMS
                reset!(solution, nsize)
                random_connect_type!(input, solution, nsize)
                evaluate!(input, solution, nsize)
                update_memory!(solution, memory)
                next!(progress)
            end
        end
    end
end
for iter = 1:ITER
    reset!(solution, nsize)
    improvise_search!(input, solution, memory, nsize, HMCR, PAR)
    evaluate!(input, solution, nsize)
    if solution.is_feasible
        worst_obj, worst_idx = findmax(memory.total_obj)
        if solution.total_obj < worst_obj
            remove_from_memory!(memory, worst_idx)
        update_memory!(solution, memory)
        end
    end
    next!(progress)
end

stoptime = time()
processtime = round(stoptime - startime, 2)

best_obj, best_idx = findmin(memory.total_obj)
if best_obj < best.total_obj
    update_best!(best, memory, best_idx, processtime)
end

variant_file = string("../output/n", nsize, "/instance_", nsize, "-", variant, ".txt")
vf = open(variant_file, "w")
write_best(input, best, nsize, vf)
close(vf)
write(of, string(variant, " ", best.total_obj, " ", best.time, "\n"))
close(of)
end
nothing
end

# The main algorithm initializer
main()
module Memory
export HarmonyMemory

type HarmonyMemory
    connect_type::Vector{Matrix{Int}}
    distance::Vector{Matrix{Int}}
    capacity::Vector{Matrix{Int}}
    trans_obj::Vector{Int}
    trans_memory::Vector{Matrix{Int}}
    fixed_obj::Vector{Int}
    ap_location::Vector{Vector{Int}}
    total_obj::Vector{Int}

    function HarmonyMemory()
        connect_type = Array[]
        distance = Array[]
        capacity = Array[]
        trans_obj = Int[]
        trans_memory = Array[]
        fixed_obj = Int[]
        ap_location = Array[]
        total_obj = Int[]

        new(connect_type,
            distance,
            capacity,
            trans_obj,
            trans_memory,
            fixed_obj,
            ap_location,
            total_obj)
    end
end
end
module Solution
export PotentialSolution

type PotentialSolution
    connect_type::Matrix{Int}
distance::Matrix{Int}
capacity::Matrix{Int}
trans_obj::Int
trans_memory::Matrix{Int}
fixed_obj::Int
ap_location::Vector{Int}
total_obj::Int
is_feasible::Bool
end

function PotentialSolution(nsize::Int)
    connect_type = diagm([-1 for _ = 1:nsize])
distance = zeros(connect_type)
capacity = zeros(connect_type)
trans_obj = 0
trans_memory = zeros(connect_type)
fixed_obj = 0
ap_location = zeros(Int, nsize)
total_obj = 0
is_feasible = false
new(connect_type, distance, capacity, trans_obj, trans_memory, fixed_obj, ap_location, total_obj, is_feasible)
end
end

Appendix A-9
Proposed Algorithm Code – solution.jl
function update_memory!(solution::PotentialSolution, memory::HarmonyMemory)
    push!(memory.connect_type, copy(solution.connect_type))
    push!(memory.distance, copy(solution.distance))
    push!(memory.capacity, copy(solution.capacity))
    push!(memory.trans_obj, copy(solution.trans_obj))
    push!(memory.trans_memory, copy(solution.trans_memory))
    push!(memory.fixed_obj, copy(solution.fixed_obj))
    push!(memory.ap_location, copy(solution.ap_location))
    push!(memory.total_obj, copy(solution.total_obj))
    nothing
end

function remove_from_memory!(memory::HarmonyMemory, idx::Int)
    deleteat!(memory.connect_type, idx)
    deleteat!(memory.distance, idx)
    deleteat!(memory.capacity, idx)
    deleteat!(memory.trans_obj, idx)
    deleteat!(memory.trans_memory, idx)
    deleteat!(memory.fixed_obj, idx)
    deleteat!(memory.ap_location, idx)
    deleteat!(memory.total_obj, idx)
    nothing
end

function reset!(solution::PotentialSolution, nsize::Int)
    for j=1:nsize
        solution.ap_location[j] = 0
    end
    for i=j+1:nsize
        solution.distance[i,j] = 0
        solution.distance[j,i] = 0
        solution.capacity[i,j] = 0
        solution.capacity[j,i] = 0
        solution.trans_memory[i,j] = 0
        solution.trans_memory[j,i] = 0
    end
end
function write_best(data::InputData, best::Best, nsize::Int, f)
    write(f, string("Process Time ==> ", best.time, " seconds\n"))
    write(f, string("Total Cost ==> ", best.total_obj, "\n"))
    write(f, string("Fixed Cost ==> ", best.fixed_obj, "\n"))
    write(f, string("AP ::\n"))
    for i = 1:nsize
        if best.ap_location[i] > 0
            write(f, string("x [ ", i, " ] = ", best.ap_location[i], "\t\t", data.ap_cost[i], "\n"))
        end
    end
    write(f, string("LAN ::\n"))
    for i = 1:nsize, j = 1:nsize
        if best.connect_type[i, j] == 1 && best.trans_memory[i, j] > 0
            write(f, string("z [ ", i, " ][ ", j, " ] = 1\t", data.lan_cost[i, j], "\n"))
        end
    end
    for i = 1:nsize, j = 1:nsize
        if best.connect_type[i, j] == 3 && best.trans_memory[i, j] > 0
            write(f, string("w [ ", i, " ][ ", j, " ] = 1\n"))
        end
    end
    write(f, string("Transfer Cost ==> ", best.trans_obj, "\n"))
    for i = 1:nsize, j = 1:nsize
        if best.trans_memory[i, j] > 0 && best.connect_type[i, j] == 1
            write(f, string("y [ ", i, " ][ ", j, " ] = ", best.trans_memory[i, j], "\t\t", *(best.trans_memory[i, j], data.recti[i, j], data.lan_transfer_cost), "\n"))
        end
    end
end

write(f, string("v [ ", i, " ] [ ", j, " ] = ", best.trans_memory[i, j]))
write(f, string("\t\t", *(best.trans_memory[i, j], data.eucli[i, j], data.wlan_transfer_cost), "\n"))
end
end
nothing
end

function update_best!(best::Best, memory::HarmonyMemory, idx::Int, time::Float64)
    best.time = time
    best.total_obj = memory.total_obj[idx]
    best.fixed_obj = memory.fixed_obj[idx]
    best.trans_obj = memory.trans_obj[idx]
    best.trans_memory = memory.trans_memory[idx]
    best.ap_location = memory.ap_location[idx]
    best.connect_type = memory.connect_type[idx]
end
APPENDIX B

EXAMPLE OF DATA AND NETWORK RESULT TEXT FILE

Appendix B shows the example of the content inside the text file of the instance data (input) and network result (output) of the proposed algorithm. The example illustrates by using node size of 20 with data structure (A and B).
Appendix B-1
Instance Data Text File – Structure A Node Size of 20

xyz
0 1 0
10 0 5
5 2 7
7 0 6
6 7 7
3 9 5
10 9 3
3 10 6
9 2 1
5 5 9
9 9 3
9 3 3
9 4 6
8 7 0
3 3 7
5 6 9
6 6 7
9 8 2
10 5 8
5 5 4
demand
0 1 2 1 6 0 10 4 9 5 10 0 8 9 1 8 0 0 0 7
ap_cost
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
lan_cap
0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5 5
5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 0 5 5 5
wlan_cap
0 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5

66
lan_transfer_cost 1
wlan_transfer_cost 1
max_wlan_distance 5
Appendix B-2

Instance Data Text File – Structure B Node Size of 20

xyz
9 30 95
22 90 33
28 99 49
40 95 88
8 56 79
19 77 52
89 47 16
60 100 44
75 83 84
32 21 27
56 56 1
31 93 74
38 29 38
21 89 62
43 24 55
53 10 93
67 76 43
23 81 98
11 49 12
83 43 32
demand
0 10 6 4 5 6 3 9 4 7 9 5 0 5 0 8 0 3 10 5
ap_cost
200 200 200 200 200 200 200 200 200 200 200 200 200 200 200
200 200 200 200 200
lan_cap
0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 0 6 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 0 6 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 0 6 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 0 6 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 0 6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 0 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 0 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 0 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 0 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 6 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 6 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 6 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0 6
6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 0
wlan_cap
lan_transfer_cost
1
wlan_transfer_cost
1
max_wlan_distance
60
Appendix B-3

Network Result Text File – Node Size of 20

Process Time ==> 4.84 seconds
Total Cost ==> 4536
Fixed Cost ==> 3120

AP ::

x [ 1 ] = 1  20
x [ 2 ] = 1  20
x [ 3 ] = 1  20
x [ 4 ] = 1  20
x [ 5 ] = 1  20
x [ 6 ] = 1  20
x [ 7 ] = 1  20
x [ 8 ] = 1  20
x [ 9 ] = 1  20
x [10 ] = 1  20
x [12 ] = 1  20
x [13 ] = 1  20
x [14 ] = 1  20
x [15 ] = 1  20
x [16 ] = 1  20
x [17 ] = 1  20
x [18 ] = 1  20
x [20 ] = 1  20

LAN ::

z [ 1 ][ 2 ] = 1  160
z [ 1 ][ 3 ] = 1  130
z [ 1 ][ 4 ] = 1  140
z [ 1 ][ 5 ] = 1  190
z [ 1 ][ 6 ] = 1  160
z [ 1 ][ 7 ] = 1  210
z [ 1 ][ 9 ] = 1  110
z [ 1 ][10 ] = 1
z [ 1 ][11 ] = 1  200
z [ 1 ][12 ] = 1  140
z [ 1 ][13 ] = 1  180
z [ 1 ][14 ] = 1  140
z [ 1 ][15 ] = 1  120
z [ 1 ][16 ] = 1  190
z [ 1 ][17 ] = 1  180
z [ 1 ][18 ] = 1  180
z [ 1 ][20 ] = 1  130
w [ 2 ][12 ] = 1
w [ 3 ][10 ] = 1
w [ 4 ][13 ] = 1
w [ 6 ][ 8 ] = 1
w [10 ][16 ] = 1
w [12 ][ 9 ] = 1
w [12 ][14 ] = 1
\( w [ 13 ][ 11 ] = 1 \)
\( w [ 15 ][ 20 ] = 1 \)
\( w [ 17 ][ 5 ] = 1 \)
\( w [ 17 ][ 11 ] = 1 \)
\( w [ 18 ][ 7 ] = 1 \)
\( w [ 20 ][ 14 ] = 1 \)

Transfer Cost ==> 1416

\( y [ 1 ][ 2 ] = 2 \) 32
\( y [ 1 ][ 3 ] = 5 \) 65
\( y [ 1 ][ 4 ] = 5 \) 70
\( y [ 1 ][ 5 ] = 5 \) 95
\( y [ 1 ][ 6 ] = 4 \) 64
\( y [ 1 ][ 7 ] = 5 \) 105
\( y [ 1 ][ 9 ] = 5 \) 55
\( y [ 1 ][ 10 ] = 5 \) 90
\( y [ 1 ][ 11 ] = 5 \) 100
\( y [ 1 ][ 12 ] = 5 \) 70
\( y [ 1 ][ 13 ] = 5 \) 90
\( y [ 1 ][ 14 ] = 5 \) 70
\( y [ 1 ][ 15 ] = 5 \) 60
\( y [ 1 ][ 16 ] = 5 \) 95
\( y [ 1 ][ 17 ] = 5 \) 90
\( y [ 1 ][ 18 ] = 5 \) 90
\( y [ 1 ][ 20 ] = 5 \) 65
\( v [ 2 ][ 12 ] = 1 \) 3
\( v [ 3 ][ 10 ] = 3 \) 9
\( v [ 4 ][ 13 ] = 4 \) 16
\( v [ 6 ][ 8 ] = 4 \) 4
\( v [ 10 ][ 16 ] = 3 \) 3
\( v [ 12 ][ 9 ] = 4 \) 8
\( v [ 12 ][ 14 ] = 2 \) 10
\( v [ 13 ][ 11 ] = 1 \) 5
\( v [ 15 ][ 20 ] = 4 \) 16
\( v [ 17 ][ 5 ] = 1 \) 1
\( v [ 17 ][ 11 ] = 4 \) 20
\( v [ 18 ][ 7 ] = 5 \) 5
\( v [ 20 ][ 14 ] = 2 \) 10
Appendix C shows the CPLEX, LS1, LS2, and LS3 objective value and runtime result tables. The data inside these tables are used for analysis and evaluation as discussed in Chapter 7. Noting that CPLEX cannot solve some instances starting at the node size of 70, which means that the data of objective value and runtime will be missing (indicated by “-” sign).
## Appendix C-1

### CPLEX Objective Value Table

<table>
<thead>
<tr>
<th></th>
<th>CPLEX obj</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4520</td>
</tr>
<tr>
<td>2</td>
<td>2575</td>
</tr>
<tr>
<td>3</td>
<td>1900</td>
</tr>
<tr>
<td>4</td>
<td>3050</td>
</tr>
<tr>
<td>5</td>
<td>2946</td>
</tr>
<tr>
<td>6</td>
<td>2555</td>
</tr>
<tr>
<td>7</td>
<td>2370</td>
</tr>
<tr>
<td>8</td>
<td>2755</td>
</tr>
<tr>
<td>9</td>
<td>3691</td>
</tr>
<tr>
<td>10</td>
<td>3604</td>
</tr>
<tr>
<td>11</td>
<td>2607</td>
</tr>
<tr>
<td>12</td>
<td>2575</td>
</tr>
<tr>
<td>13</td>
<td>3500</td>
</tr>
<tr>
<td>14</td>
<td>2707</td>
</tr>
<tr>
<td>15</td>
<td>4310</td>
</tr>
<tr>
<td>16</td>
<td>2573</td>
</tr>
<tr>
<td>17</td>
<td>2503</td>
</tr>
<tr>
<td>18</td>
<td>4369</td>
</tr>
<tr>
<td>19</td>
<td>3069</td>
</tr>
<tr>
<td>20</td>
<td>1178</td>
</tr>
<tr>
<td>21</td>
<td>30950</td>
</tr>
<tr>
<td>22</td>
<td>20234</td>
</tr>
<tr>
<td>23</td>
<td>27827</td>
</tr>
<tr>
<td>24</td>
<td>11004</td>
</tr>
<tr>
<td>25</td>
<td>32022</td>
</tr>
<tr>
<td>26</td>
<td>13558</td>
</tr>
<tr>
<td>27</td>
<td>19022</td>
</tr>
<tr>
<td>28</td>
<td>21140</td>
</tr>
<tr>
<td>29</td>
<td>27025</td>
</tr>
<tr>
<td>30</td>
<td>18405</td>
</tr>
<tr>
<td>31</td>
<td>22719</td>
</tr>
<tr>
<td>32</td>
<td>22159</td>
</tr>
<tr>
<td>33</td>
<td>27084</td>
</tr>
<tr>
<td>34</td>
<td>12751</td>
</tr>
<tr>
<td>35</td>
<td>8933</td>
</tr>
<tr>
<td>36</td>
<td>21522</td>
</tr>
<tr>
<td>37</td>
<td>30347</td>
</tr>
<tr>
<td>38</td>
<td>31341</td>
</tr>
<tr>
<td>39</td>
<td>40091</td>
</tr>
<tr>
<td>40</td>
<td>21522</td>
</tr>
</tbody>
</table>
## Appendix C-2

### CPLEX Runtime Table

<table>
<thead>
<tr>
<th>CPLEX time</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.78</td>
<td>2.34</td>
<td>3.84</td>
<td>12.90</td>
<td>22.57</td>
<td>36.75</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>0.38</td>
<td>1.19</td>
<td>2.48</td>
<td>11.90</td>
<td>34.55</td>
<td>12.03</td>
<td>115.68</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>0.45</td>
<td>0.82</td>
<td>2.54</td>
<td>5.21</td>
<td>25.99</td>
<td>148.74</td>
<td>43.90</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>4</td>
<td>0.36</td>
<td>1.95</td>
<td>1.97</td>
<td>10.50</td>
<td>19.09</td>
<td>77.33</td>
<td>60.75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>5</td>
<td>0.62</td>
<td>1.86</td>
<td>3.48</td>
<td>8.80</td>
<td>36.41</td>
<td>23.34</td>
<td>28.81</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>6</td>
<td>0.34</td>
<td>0.91</td>
<td>4.32</td>
<td>12.40</td>
<td>13.90</td>
<td>101.94</td>
<td>313.38</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>0.76</td>
<td>2.01</td>
<td>1.83</td>
<td>13.98</td>
<td>17.30</td>
<td>29.92</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>8</td>
<td>0.34</td>
<td>1.89</td>
<td>6.02</td>
<td>12.79</td>
<td>6.10</td>
<td>303.43</td>
<td>60.42</td>
<td>73.30</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>0.58</td>
<td>0.73</td>
<td>5.54</td>
<td>4.51</td>
<td>18.71</td>
<td>45.02</td>
<td>25.91</td>
<td>41.14</td>
<td>-</td>
</tr>
<tr>
<td>10</td>
<td>0.42</td>
<td>1.40</td>
<td>5.21</td>
<td>14.76</td>
<td>61.53</td>
<td>87.58</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>11</td>
<td>0.53</td>
<td>1.37</td>
<td>3.17</td>
<td>11.47</td>
<td>69.20</td>
<td>13.70</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>12</td>
<td>0.63</td>
<td>1.17</td>
<td>3.87</td>
<td>12.08</td>
<td>19.31</td>
<td>25.58</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>13</td>
<td>0.33</td>
<td>1.78</td>
<td>4.40</td>
<td>44.87</td>
<td>13.73</td>
<td>26.71</td>
<td>16.27</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>14</td>
<td>0.78</td>
<td>1.62</td>
<td>4.65</td>
<td>10.52</td>
<td>18.36</td>
<td>142.31</td>
<td>-</td>
<td>26.26</td>
<td>-</td>
</tr>
<tr>
<td>15</td>
<td>0.28</td>
<td>3.31</td>
<td>5.41</td>
<td>14.93</td>
<td>50.89</td>
<td>12.23</td>
<td>46.25</td>
<td>21.27</td>
<td>-</td>
</tr>
<tr>
<td>16</td>
<td>0.76</td>
<td>1.03</td>
<td>2.31</td>
<td>10.64</td>
<td>56.13</td>
<td>27.50</td>
<td>445.14</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>17</td>
<td>0.69</td>
<td>2.23</td>
<td>4.01</td>
<td>11.39</td>
<td>10.69</td>
<td>30.76</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>18</td>
<td>0.39</td>
<td>2.84</td>
<td>6.24</td>
<td>14.79</td>
<td>14.76</td>
<td>37.07</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>19</td>
<td>0.34</td>
<td>0.73</td>
<td>2.22</td>
<td>8.07</td>
<td>26.74</td>
<td>62.06</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>20</td>
<td>1.01</td>
<td>2.20</td>
<td>3.56</td>
<td>13.90</td>
<td>14.13</td>
<td>178.93</td>
<td>79.95</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>21</td>
<td>0.36</td>
<td>1.95</td>
<td>5.73</td>
<td>15.66</td>
<td>17.93</td>
<td>21.62</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>22</td>
<td>0.27</td>
<td>1.89</td>
<td>2.78</td>
<td>92.04</td>
<td>17.18</td>
<td>29.53</td>
<td>60.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>23</td>
<td>0.41</td>
<td>2.75</td>
<td>3.48</td>
<td>4.93</td>
<td>25.99</td>
<td>13.37</td>
<td>42.95</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>24</td>
<td>0.69</td>
<td>1.33</td>
<td>2.57</td>
<td>9.95</td>
<td>7.52</td>
<td>47.64</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>25</td>
<td>0.76</td>
<td>1.59</td>
<td>3.76</td>
<td>5.07</td>
<td>20.87</td>
<td>63.90</td>
<td>43.06</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>26</td>
<td>0.33</td>
<td>2.01</td>
<td>5.29</td>
<td>5.84</td>
<td>14.23</td>
<td>27.80</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>27</td>
<td>0.34</td>
<td>1.48</td>
<td>4.21</td>
<td>9.13</td>
<td>23.29</td>
<td>31.62</td>
<td>24.01</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>28</td>
<td>0.37</td>
<td>2.29</td>
<td>2.92</td>
<td>10.59</td>
<td>45.54</td>
<td>45.19</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>29</td>
<td>0.80</td>
<td>1.17</td>
<td>1.98</td>
<td>10.28</td>
<td>28.53</td>
<td>28.69</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>30</td>
<td>0.28</td>
<td>2.67</td>
<td>4.20</td>
<td>10.86</td>
<td>19.58</td>
<td>14.20</td>
<td>79.31</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>31</td>
<td>0.36</td>
<td>2.09</td>
<td>4.96</td>
<td>10.61</td>
<td>24.84</td>
<td>14.85</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>32</td>
<td>0.38</td>
<td>2.33</td>
<td>2.53</td>
<td>8.99</td>
<td>11.89</td>
<td>30.14</td>
<td>49.73</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>33</td>
<td>0.86</td>
<td>2.09</td>
<td>6.40</td>
<td>14.57</td>
<td>16.29</td>
<td>38.44</td>
<td>73.09</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>34</td>
<td>0.45</td>
<td>2.65</td>
<td>3.67</td>
<td>19.11</td>
<td>15.74</td>
<td>28.53</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>35</td>
<td>0.30</td>
<td>1.00</td>
<td>5.99</td>
<td>13.63</td>
<td>47.78</td>
<td>13.76</td>
<td>142.71</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>36</td>
<td>0.76</td>
<td>2.92</td>
<td>6.15</td>
<td>10.70</td>
<td>26.21</td>
<td>15.76</td>
<td>49.27</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>37</td>
<td>1.17</td>
<td>1.86</td>
<td>4.81</td>
<td>12.23</td>
<td>6.86</td>
<td>14.41</td>
<td>45.04</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>38</td>
<td>0.33</td>
<td>1.05</td>
<td>4.57</td>
<td>5.97</td>
<td>29.30</td>
<td>35.13</td>
<td>332.62</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>39</td>
<td>0.80</td>
<td>0.78</td>
<td>5.13</td>
<td>7.80</td>
<td>19.16</td>
<td>35.29</td>
<td>56.61</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>40</td>
<td>0.34</td>
<td>2.09</td>
<td>7.21</td>
<td>5.07</td>
<td>19.27</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Appendix C-3

LS1 Objective Value Table

<table>
<thead>
<tr>
<th></th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4528</td>
<td>4944</td>
<td>6687</td>
<td>4591</td>
<td>10476</td>
<td>11781</td>
<td>11177</td>
<td>11043</td>
<td>15951</td>
</tr>
<tr>
<td>2</td>
<td>2763</td>
<td>6538</td>
<td>7866</td>
<td>6437</td>
<td>7866</td>
<td>11120</td>
<td>10472</td>
<td>12958</td>
<td>17909</td>
</tr>
<tr>
<td>3</td>
<td>2035</td>
<td>5235</td>
<td>7864</td>
<td>8861</td>
<td>6478</td>
<td>9784</td>
<td>8488</td>
<td>9618</td>
<td>13086</td>
</tr>
<tr>
<td>4</td>
<td>3295</td>
<td>4858</td>
<td>5051</td>
<td>7923</td>
<td>4791</td>
<td>11411</td>
<td>14046</td>
<td>9453</td>
<td>14125</td>
</tr>
<tr>
<td>5</td>
<td>2946</td>
<td>2924</td>
<td>5561</td>
<td>7436</td>
<td>6138</td>
<td>12749</td>
<td>15029</td>
<td>10675</td>
<td>12756</td>
</tr>
<tr>
<td>6</td>
<td>2583</td>
<td>4179</td>
<td>5787</td>
<td>8467</td>
<td>5843</td>
<td>17125</td>
<td>12605</td>
<td>13135</td>
<td>11207</td>
</tr>
<tr>
<td>7</td>
<td>2388</td>
<td>4656</td>
<td>6350</td>
<td>6497</td>
<td>10487</td>
<td>8545</td>
<td>11737</td>
<td>9277</td>
<td>15138</td>
</tr>
<tr>
<td>8</td>
<td>2766</td>
<td>4396</td>
<td>4508</td>
<td>6725</td>
<td>8255</td>
<td>9330</td>
<td>8194</td>
<td>13323</td>
<td>15973</td>
</tr>
<tr>
<td>9</td>
<td>3730</td>
<td>3261</td>
<td>5752</td>
<td>6936</td>
<td>9448</td>
<td>13049</td>
<td>7340</td>
<td>12592</td>
<td>13958</td>
</tr>
<tr>
<td>10</td>
<td>3626</td>
<td>2886</td>
<td>6861</td>
<td>5020</td>
<td>8753</td>
<td>10406</td>
<td>9406</td>
<td>13672</td>
<td>14641</td>
</tr>
<tr>
<td>11</td>
<td>2816</td>
<td>3805</td>
<td>4700</td>
<td>4609</td>
<td>8536</td>
<td>6680</td>
<td>11752</td>
<td>16183</td>
<td>19597</td>
</tr>
<tr>
<td>12</td>
<td>2654</td>
<td>5050</td>
<td>7293</td>
<td>6366</td>
<td>10500</td>
<td>10554</td>
<td>11757</td>
<td>15715</td>
<td>17240</td>
</tr>
<tr>
<td>13</td>
<td>3682</td>
<td>4709</td>
<td>4233</td>
<td>6945</td>
<td>8035</td>
<td>9430</td>
<td>9912</td>
<td>11467</td>
<td>10385</td>
</tr>
<tr>
<td>14</td>
<td>2976</td>
<td>4068</td>
<td>7090</td>
<td>9465</td>
<td>8371</td>
<td>11500</td>
<td>11250</td>
<td>13300</td>
<td>14119</td>
</tr>
<tr>
<td>15</td>
<td>4391</td>
<td>4639</td>
<td>6217</td>
<td>4639</td>
<td>9029</td>
<td>11711</td>
<td>13239</td>
<td>10050</td>
<td>15675</td>
</tr>
<tr>
<td>16</td>
<td>2592</td>
<td>3074</td>
<td>5407</td>
<td>8144</td>
<td>8789</td>
<td>10852</td>
<td>9761</td>
<td>15339</td>
<td>17549</td>
</tr>
<tr>
<td>17</td>
<td>2766</td>
<td>3438</td>
<td>6231</td>
<td>6148</td>
<td>7961</td>
<td>11253</td>
<td>12779</td>
<td>15733</td>
<td>15702</td>
</tr>
<tr>
<td>18</td>
<td>4530</td>
<td>3400</td>
<td>4400</td>
<td>6371</td>
<td>6174</td>
<td>10496</td>
<td>7501</td>
<td>14053</td>
<td>15272</td>
</tr>
<tr>
<td>19</td>
<td>3072</td>
<td>3753</td>
<td>6840</td>
<td>6110</td>
<td>5001</td>
<td>9776</td>
<td>11500</td>
<td>12393</td>
<td>13592</td>
</tr>
<tr>
<td>20</td>
<td>1268</td>
<td>4455</td>
<td>7141</td>
<td>7492</td>
<td>9817</td>
<td>11216</td>
<td>14500</td>
<td>9417</td>
<td>16213</td>
</tr>
<tr>
<td>21</td>
<td>31294</td>
<td>33602</td>
<td>61380</td>
<td>57458</td>
<td>101773</td>
<td>101949</td>
<td>137505</td>
<td>73674</td>
<td>139624</td>
</tr>
<tr>
<td>22</td>
<td>20642</td>
<td>54339</td>
<td>49186</td>
<td>55457</td>
<td>74732</td>
<td>81744</td>
<td>86623</td>
<td>164371</td>
<td>90968</td>
</tr>
<tr>
<td>23</td>
<td>28199</td>
<td>53262</td>
<td>40668</td>
<td>90667</td>
<td>46333</td>
<td>98513</td>
<td>88988</td>
<td>95559</td>
<td>112638</td>
</tr>
<tr>
<td>24</td>
<td>12163</td>
<td>18421</td>
<td>63237</td>
<td>81326</td>
<td>33444</td>
<td>89143</td>
<td>97925</td>
<td>103704</td>
<td>112406</td>
</tr>
<tr>
<td>25</td>
<td>32259</td>
<td>43043</td>
<td>45640</td>
<td>65907</td>
<td>74332</td>
<td>92589</td>
<td>79649</td>
<td>105702</td>
<td>131279</td>
</tr>
<tr>
<td>26</td>
<td>15046</td>
<td>50103</td>
<td>48538</td>
<td>36686</td>
<td>57859</td>
<td>66234</td>
<td>90834</td>
<td>92269</td>
<td>133017</td>
</tr>
<tr>
<td>27</td>
<td>19534</td>
<td>38403</td>
<td>60425</td>
<td>53512</td>
<td>88230</td>
<td>86606</td>
<td>120334</td>
<td>117070</td>
<td>71418</td>
</tr>
<tr>
<td>28</td>
<td>21235</td>
<td>39913</td>
<td>50517</td>
<td>46748</td>
<td>105459</td>
<td>84252</td>
<td>125425</td>
<td>99249</td>
<td>148872</td>
</tr>
<tr>
<td>29</td>
<td>27072</td>
<td>23273</td>
<td>57218</td>
<td>62099</td>
<td>46802</td>
<td>85540</td>
<td>171285</td>
<td>130864</td>
<td>130565</td>
</tr>
<tr>
<td>30</td>
<td>20238</td>
<td>31432</td>
<td>71434</td>
<td>51265</td>
<td>46313</td>
<td>102919</td>
<td>119574</td>
<td>71401</td>
<td>140868</td>
</tr>
<tr>
<td>31</td>
<td>23126</td>
<td>34159</td>
<td>44839</td>
<td>61555</td>
<td>95544</td>
<td>95965</td>
<td>74445</td>
<td>112984</td>
<td>125248</td>
</tr>
<tr>
<td>32</td>
<td>22537</td>
<td>43926</td>
<td>32369</td>
<td>83319</td>
<td>69098</td>
<td>87936</td>
<td>90685</td>
<td>104644</td>
<td>124723</td>
</tr>
<tr>
<td>33</td>
<td>27582</td>
<td>47779</td>
<td>58142</td>
<td>65252</td>
<td>62321</td>
<td>70023</td>
<td>113073</td>
<td>134747</td>
<td>105213</td>
</tr>
<tr>
<td>34</td>
<td>12947</td>
<td>28366</td>
<td>47861</td>
<td>54258</td>
<td>91343</td>
<td>140607</td>
<td>85391</td>
<td>118657</td>
<td>108054</td>
</tr>
<tr>
<td>35</td>
<td>9197</td>
<td>27224</td>
<td>42300</td>
<td>48305</td>
<td>86070</td>
<td>73481</td>
<td>101021</td>
<td>83292</td>
<td>134565</td>
</tr>
<tr>
<td>36</td>
<td>22153</td>
<td>40451</td>
<td>77662</td>
<td>53889</td>
<td>70523</td>
<td>102561</td>
<td>79508</td>
<td>133159</td>
<td>196554</td>
</tr>
<tr>
<td>37</td>
<td>30507</td>
<td>25122</td>
<td>45781</td>
<td>68712</td>
<td>36540</td>
<td>47521</td>
<td>67927</td>
<td>101569</td>
<td>120378</td>
</tr>
<tr>
<td>38</td>
<td>31923</td>
<td>24748</td>
<td>47116</td>
<td>44512</td>
<td>52036</td>
<td>86011</td>
<td>106501</td>
<td>104970</td>
<td>116526</td>
</tr>
<tr>
<td>39</td>
<td>40099</td>
<td>51207</td>
<td>73065</td>
<td>114774</td>
<td>82186</td>
<td>87871</td>
<td>68552</td>
<td>122230</td>
<td>144263</td>
</tr>
<tr>
<td>40</td>
<td>21679</td>
<td>36563</td>
<td>27054</td>
<td>88871</td>
<td>60347</td>
<td>49518</td>
<td>89316</td>
<td>86387</td>
<td>148003</td>
</tr>
</tbody>
</table>
## Appendix C-4

### LS1 Runtime Table

<table>
<thead>
<tr>
<th>LS1 time</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.72</td>
<td>1.25</td>
<td>1.79</td>
<td>2.42</td>
<td>4.37</td>
<td>6.3</td>
<td>9.12</td>
<td>10.33</td>
<td>13.45</td>
</tr>
<tr>
<td>2</td>
<td>0.44</td>
<td>0.97</td>
<td>1.9</td>
<td>2.75</td>
<td>3.96</td>
<td>6.32</td>
<td>8.32</td>
<td>10.58</td>
<td>15.05</td>
</tr>
<tr>
<td>3</td>
<td>0.42</td>
<td>1.05</td>
<td>1.86</td>
<td>2.78</td>
<td>3.81</td>
<td>5.77</td>
<td>7.68</td>
<td>10.2</td>
<td>14.09</td>
</tr>
<tr>
<td>4</td>
<td>0.44</td>
<td>0.97</td>
<td>1.62</td>
<td>2.67</td>
<td>3.59</td>
<td>6.74</td>
<td>9.7</td>
<td>10.36</td>
<td>15.6</td>
</tr>
<tr>
<td>5</td>
<td>0.44</td>
<td>0.89</td>
<td>1.89</td>
<td>2.65</td>
<td>3.48</td>
<td>6.26</td>
<td>9.23</td>
<td>11.45</td>
<td>13.74</td>
</tr>
<tr>
<td>6</td>
<td>0.47</td>
<td>0.84</td>
<td>1.78</td>
<td>2.89</td>
<td>3.78</td>
<td>5.1</td>
<td>8.06</td>
<td>11.34</td>
<td>13.9</td>
</tr>
<tr>
<td>7</td>
<td>0.44</td>
<td>0.95</td>
<td>1.78</td>
<td>2.68</td>
<td>4.45</td>
<td>5.88</td>
<td>8.16</td>
<td>9.91</td>
<td>13.85</td>
</tr>
<tr>
<td>8</td>
<td>0.47</td>
<td>0.89</td>
<td>1.67</td>
<td>2.79</td>
<td>3.87</td>
<td>5.93</td>
<td>7.73</td>
<td>11.61</td>
<td>14.82</td>
</tr>
<tr>
<td>9</td>
<td>0.47</td>
<td>0.9</td>
<td>1.76</td>
<td>2.71</td>
<td>4.35</td>
<td>6.71</td>
<td>7.33</td>
<td>11.64</td>
<td>14.23</td>
</tr>
<tr>
<td>10</td>
<td>0.45</td>
<td>0.83</td>
<td>1.78</td>
<td>2.71</td>
<td>4.04</td>
<td>5.96</td>
<td>7.91</td>
<td>11.5</td>
<td>13.68</td>
</tr>
<tr>
<td>11</td>
<td>0.45</td>
<td>0.8</td>
<td>1.47</td>
<td>2.57</td>
<td>3.9</td>
<td>5.38</td>
<td>8.61</td>
<td>11.98</td>
<td>15.71</td>
</tr>
<tr>
<td>12</td>
<td>0.42</td>
<td>0.95</td>
<td>1.73</td>
<td>2.42</td>
<td>4.38</td>
<td>6.52</td>
<td>9.3</td>
<td>13.46</td>
<td>14.71</td>
</tr>
<tr>
<td>13</td>
<td>0.5</td>
<td>0.92</td>
<td>1.59</td>
<td>2.42</td>
<td>3.99</td>
<td>6.72</td>
<td>8.02</td>
<td>11.33</td>
<td>14.07</td>
</tr>
<tr>
<td>14</td>
<td>0.42</td>
<td>0.9</td>
<td>1.78</td>
<td>3.59</td>
<td>4.2</td>
<td>5.8</td>
<td>8.64</td>
<td>10.56</td>
<td>15.01</td>
</tr>
<tr>
<td>15</td>
<td>0.47</td>
<td>0.98</td>
<td>1.92</td>
<td>2.36</td>
<td>3.71</td>
<td>6.51</td>
<td>9.06</td>
<td>11.25</td>
<td>15.54</td>
</tr>
<tr>
<td>16</td>
<td>0.44</td>
<td>0.83</td>
<td>1.59</td>
<td>2.85</td>
<td>4.52</td>
<td>6.04</td>
<td>8.55</td>
<td>12.56</td>
<td>14.87</td>
</tr>
<tr>
<td>17</td>
<td>0.42</td>
<td>0.86</td>
<td>1.7</td>
<td>2.61</td>
<td>4.1</td>
<td>6.49</td>
<td>8.64</td>
<td>11.4</td>
<td>14.26</td>
</tr>
<tr>
<td>18</td>
<td>0.4</td>
<td>0.84</td>
<td>1.62</td>
<td>2.7</td>
<td>3.68</td>
<td>5.66</td>
<td>7.83</td>
<td>12.25</td>
<td>13.93</td>
</tr>
<tr>
<td>19</td>
<td>0.45</td>
<td>0.78</td>
<td>1.9</td>
<td>2.4</td>
<td>3.81</td>
<td>5.87</td>
<td>8.1</td>
<td>10.55</td>
<td>13.85</td>
</tr>
<tr>
<td>20</td>
<td>0.42</td>
<td>0.92</td>
<td>1.75</td>
<td>2.68</td>
<td>4.4</td>
<td>6.57</td>
<td>9.16</td>
<td>11.03</td>
<td>14.96</td>
</tr>
<tr>
<td>21</td>
<td>0.44</td>
<td>0.97</td>
<td>1.78</td>
<td>2.65</td>
<td>4.37</td>
<td>6.91</td>
<td>9.53</td>
<td>9.78</td>
<td>14.76</td>
</tr>
<tr>
<td>22</td>
<td>0.45</td>
<td>1</td>
<td>1.67</td>
<td>2.56</td>
<td>4.48</td>
<td>6.4</td>
<td>8.61</td>
<td>11.69</td>
<td>12.59</td>
</tr>
<tr>
<td>23</td>
<td>0.42</td>
<td>0.92</td>
<td>1.68</td>
<td>2.96</td>
<td>3.67</td>
<td>6.22</td>
<td>8.22</td>
<td>10.25</td>
<td>13.99</td>
</tr>
<tr>
<td>24</td>
<td>0.41</td>
<td>0.86</td>
<td>1.95</td>
<td>2.92</td>
<td>3.37</td>
<td>6.36</td>
<td>8</td>
<td>11.09</td>
<td>13.31</td>
</tr>
<tr>
<td>25</td>
<td>0.51</td>
<td>0.92</td>
<td>1.73</td>
<td>2.48</td>
<td>4.48</td>
<td>6.21</td>
<td>8.19</td>
<td>11.34</td>
<td>14.17</td>
</tr>
<tr>
<td>26</td>
<td>0.42</td>
<td>0.98</td>
<td>1.65</td>
<td>2.34</td>
<td>3.82</td>
<td>5.35</td>
<td>8.08</td>
<td>9.7</td>
<td>14.9</td>
</tr>
<tr>
<td>27</td>
<td>0.47</td>
<td>0.94</td>
<td>1.9</td>
<td>2.64</td>
<td>4.06</td>
<td>5.71</td>
<td>7.83</td>
<td>10.58</td>
<td>12.62</td>
</tr>
<tr>
<td>28</td>
<td>0.45</td>
<td>0.94</td>
<td>1.76</td>
<td>3.46</td>
<td>4.1</td>
<td>6.21</td>
<td>8.2</td>
<td>9.64</td>
<td>14.93</td>
</tr>
<tr>
<td>29</td>
<td>0.47</td>
<td>0.94</td>
<td>1.79</td>
<td>2.73</td>
<td>3.87</td>
<td>5.9</td>
<td>9.72</td>
<td>11.33</td>
<td>14.59</td>
</tr>
<tr>
<td>30</td>
<td>0.41</td>
<td>0.94</td>
<td>1.59</td>
<td>2.64</td>
<td>3.62</td>
<td>5.97</td>
<td>7.94</td>
<td>10.16</td>
<td>15.07</td>
</tr>
<tr>
<td>31</td>
<td>0.45</td>
<td>1</td>
<td>1.7</td>
<td>2.65</td>
<td>4.13</td>
<td>6.51</td>
<td>7.74</td>
<td>11.33</td>
<td>14.79</td>
</tr>
<tr>
<td>32</td>
<td>0.45</td>
<td>0.98</td>
<td>1.55</td>
<td>2.95</td>
<td>4.01</td>
<td>5.9</td>
<td>7.67</td>
<td>10.72</td>
<td>15.48</td>
</tr>
<tr>
<td>33</td>
<td>0.47</td>
<td>0.97</td>
<td>1.67</td>
<td>2.68</td>
<td>4.02</td>
<td>5.88</td>
<td>8.1</td>
<td>10.31</td>
<td>13.26</td>
</tr>
<tr>
<td>34</td>
<td>0.44</td>
<td>0.83</td>
<td>1.75</td>
<td>2.68</td>
<td>4.49</td>
<td>6.63</td>
<td>7.75</td>
<td>10.92</td>
<td>13.17</td>
</tr>
<tr>
<td>35</td>
<td>0.41</td>
<td>0.87</td>
<td>1.58</td>
<td>2.62</td>
<td>4.23</td>
<td>5.74</td>
<td>8.53</td>
<td>9.36</td>
<td>13.76</td>
</tr>
<tr>
<td>36</td>
<td>0.37</td>
<td>0.91</td>
<td>1.83</td>
<td>2.62</td>
<td>4.07</td>
<td>5.91</td>
<td>7.83</td>
<td>11.26</td>
<td>16.71</td>
</tr>
<tr>
<td>37</td>
<td>0.4</td>
<td>0.86</td>
<td>1.64</td>
<td>2.53</td>
<td>3.37</td>
<td>5.43</td>
<td>8.03</td>
<td>10.7</td>
<td>13.43</td>
</tr>
<tr>
<td>38</td>
<td>0.42</td>
<td>0.9</td>
<td>1.48</td>
<td>2.67</td>
<td>3.82</td>
<td>6.18</td>
<td>8.11</td>
<td>11.31</td>
<td>13.63</td>
</tr>
<tr>
<td>39</td>
<td>0.47</td>
<td>1</td>
<td>1.83</td>
<td>3.25</td>
<td>3.98</td>
<td>5.77</td>
<td>7.44</td>
<td>11</td>
<td>14.95</td>
</tr>
<tr>
<td>40</td>
<td>0.42</td>
<td>0.95</td>
<td>1.53</td>
<td>3.12</td>
<td>3.78</td>
<td>4.88</td>
<td>7.67</td>
<td>10.8</td>
<td>15.28</td>
</tr>
</tbody>
</table>
### Appendix C-5

**LS2 Objective Value Table**

<table>
<thead>
<tr>
<th>LS2 obj</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4537</td>
<td>5013</td>
<td>6574</td>
<td>4669</td>
<td>10551</td>
<td>11714</td>
<td>11118</td>
<td>10948</td>
<td>16137</td>
</tr>
<tr>
<td>2</td>
<td>2652</td>
<td>6565</td>
<td>7956</td>
<td>6440</td>
<td>7957</td>
<td>11175</td>
<td>10498</td>
<td>12986</td>
<td>18139</td>
</tr>
<tr>
<td>3</td>
<td>2037</td>
<td>5263</td>
<td>7830</td>
<td>9057</td>
<td>6581</td>
<td>9803</td>
<td>8533</td>
<td>9517</td>
<td>13017</td>
</tr>
<tr>
<td>4</td>
<td>3296</td>
<td>4957</td>
<td>5069</td>
<td>8034</td>
<td>5092</td>
<td>11561</td>
<td>14030</td>
<td>9428</td>
<td>14333</td>
</tr>
<tr>
<td>5</td>
<td>2957</td>
<td>2995</td>
<td>5574</td>
<td>7411</td>
<td>6332</td>
<td>12769</td>
<td>14960</td>
<td>10742</td>
<td>12816</td>
</tr>
<tr>
<td>6</td>
<td>2661</td>
<td>4234</td>
<td>5866</td>
<td>8349</td>
<td>5558</td>
<td>7140</td>
<td>12734</td>
<td>13011</td>
<td>11513</td>
</tr>
<tr>
<td>7</td>
<td>2406</td>
<td>4798</td>
<td>6382</td>
<td>6598</td>
<td>10436</td>
<td>8521</td>
<td>11875</td>
<td>9284</td>
<td>15259</td>
</tr>
<tr>
<td>8</td>
<td>2815</td>
<td>4483</td>
<td>4600</td>
<td>6588</td>
<td>8361</td>
<td>9395</td>
<td>8147</td>
<td>13250</td>
<td>15910</td>
</tr>
<tr>
<td>9</td>
<td>3797</td>
<td>3345</td>
<td>5833</td>
<td>6987</td>
<td>9559</td>
<td>13054</td>
<td>7360</td>
<td>12625</td>
<td>13789</td>
</tr>
<tr>
<td>10</td>
<td>3629</td>
<td>3105</td>
<td>6877</td>
<td>5141</td>
<td>8814</td>
<td>10416</td>
<td>9578</td>
<td>13805</td>
<td>14767</td>
</tr>
<tr>
<td>11</td>
<td>2839</td>
<td>3825</td>
<td>4791</td>
<td>4527</td>
<td>8650</td>
<td>6650</td>
<td>11857</td>
<td>16360</td>
<td>19708</td>
</tr>
<tr>
<td>12</td>
<td>2735</td>
<td>5118</td>
<td>7369</td>
<td>6512</td>
<td>10418</td>
<td>10803</td>
<td>11987</td>
<td>15847</td>
<td>17292</td>
</tr>
<tr>
<td>13</td>
<td>3683</td>
<td>4681</td>
<td>4248</td>
<td>6955</td>
<td>8179</td>
<td>9690</td>
<td>9877</td>
<td>11662</td>
<td>10621</td>
</tr>
<tr>
<td>14</td>
<td>2959</td>
<td>4152</td>
<td>7116</td>
<td>9564</td>
<td>8400</td>
<td>11595</td>
<td>11227</td>
<td>13258</td>
<td>14368</td>
</tr>
<tr>
<td>15</td>
<td>4433</td>
<td>4709</td>
<td>6327</td>
<td>4836</td>
<td>9166</td>
<td>11744</td>
<td>13334</td>
<td>9883</td>
<td>15935</td>
</tr>
<tr>
<td>16</td>
<td>2615</td>
<td>3076</td>
<td>5400</td>
<td>8214</td>
<td>8768</td>
<td>10936</td>
<td>9834</td>
<td>15401</td>
<td>17424</td>
</tr>
<tr>
<td>17</td>
<td>2751</td>
<td>3528</td>
<td>6302</td>
<td>6434</td>
<td>8055</td>
<td>11154</td>
<td>12741</td>
<td>15898</td>
<td>15611</td>
</tr>
<tr>
<td>18</td>
<td>4492</td>
<td>3423</td>
<td>4430</td>
<td>6239</td>
<td>6340</td>
<td>10573</td>
<td>7506</td>
<td>14129</td>
<td>15385</td>
</tr>
<tr>
<td>19</td>
<td>3074</td>
<td>3821</td>
<td>6939</td>
<td>6250</td>
<td>5028</td>
<td>9840</td>
<td>11481</td>
<td>12597</td>
<td>15372</td>
</tr>
<tr>
<td>20</td>
<td>1246</td>
<td>4430</td>
<td>7145</td>
<td>7590</td>
<td>10007</td>
<td>11376</td>
<td>14686</td>
<td>9456</td>
<td>16160</td>
</tr>
<tr>
<td>21</td>
<td>31256</td>
<td>33403</td>
<td>61526</td>
<td>57784</td>
<td>101782</td>
<td>102688</td>
<td>136472</td>
<td>72266</td>
<td>140097</td>
</tr>
<tr>
<td>22</td>
<td>20769</td>
<td>55018</td>
<td>50983</td>
<td>55899</td>
<td>74958</td>
<td>80554</td>
<td>84259</td>
<td>165131</td>
<td>92586</td>
</tr>
<tr>
<td>23</td>
<td>29024</td>
<td>52780</td>
<td>41708</td>
<td>90902</td>
<td>45570</td>
<td>96719</td>
<td>89125</td>
<td>97491</td>
<td>112017</td>
</tr>
<tr>
<td>24</td>
<td>12080</td>
<td>19762</td>
<td>63692</td>
<td>80824</td>
<td>34132</td>
<td>90049</td>
<td>96801</td>
<td>102910</td>
<td>111607</td>
</tr>
<tr>
<td>25</td>
<td>32416</td>
<td>42829</td>
<td>43983</td>
<td>67403</td>
<td>75017</td>
<td>91962</td>
<td>80043</td>
<td>104709</td>
<td>132966</td>
</tr>
<tr>
<td>26</td>
<td>14954</td>
<td>50119</td>
<td>49188</td>
<td>37325</td>
<td>58877</td>
<td>69655</td>
<td>89379</td>
<td>93045</td>
<td>134138</td>
</tr>
<tr>
<td>27</td>
<td>19675</td>
<td>38595</td>
<td>61832</td>
<td>54073</td>
<td>88513</td>
<td>88431</td>
<td>120418</td>
<td>116591</td>
<td>71916</td>
</tr>
<tr>
<td>28</td>
<td>21873</td>
<td>40290</td>
<td>51112</td>
<td>47486</td>
<td>105516</td>
<td>85259</td>
<td>124241</td>
<td>98880</td>
<td>149303</td>
</tr>
<tr>
<td>29</td>
<td>27072</td>
<td>23749</td>
<td>57361</td>
<td>63526</td>
<td>48249</td>
<td>86019</td>
<td>171362</td>
<td>130938</td>
<td>131844</td>
</tr>
<tr>
<td>30</td>
<td>20991</td>
<td>31264</td>
<td>71437</td>
<td>51382</td>
<td>49499</td>
<td>104810</td>
<td>119360</td>
<td>74019</td>
<td>142561</td>
</tr>
<tr>
<td>31</td>
<td>23930</td>
<td>34750</td>
<td>44882</td>
<td>61197</td>
<td>97863</td>
<td>96529</td>
<td>72279</td>
<td>113815</td>
<td>126951</td>
</tr>
<tr>
<td>32</td>
<td>22821</td>
<td>43526</td>
<td>33483</td>
<td>82777</td>
<td>68585</td>
<td>88575</td>
<td>90396</td>
<td>103953</td>
<td>122009</td>
</tr>
<tr>
<td>33</td>
<td>27313</td>
<td>47501</td>
<td>58732</td>
<td>68037</td>
<td>63842</td>
<td>70011</td>
<td>112083</td>
<td>136256</td>
<td>104412</td>
</tr>
<tr>
<td>34</td>
<td>13315</td>
<td>28100</td>
<td>48257</td>
<td>54612</td>
<td>90127</td>
<td>141879</td>
<td>85174</td>
<td>118992</td>
<td>107286</td>
</tr>
<tr>
<td>35</td>
<td>9993</td>
<td>27585</td>
<td>42312</td>
<td>48784</td>
<td>86048</td>
<td>76812</td>
<td>102168</td>
<td>85366</td>
<td>138438</td>
</tr>
<tr>
<td>36</td>
<td>22818</td>
<td>40261</td>
<td>78050</td>
<td>54912</td>
<td>70239</td>
<td>102363</td>
<td>79337</td>
<td>133761</td>
<td>197800</td>
</tr>
<tr>
<td>37</td>
<td>30583</td>
<td>26924</td>
<td>47624</td>
<td>69187</td>
<td>39816</td>
<td>46289</td>
<td>69060</td>
<td>101394</td>
<td>120330</td>
</tr>
<tr>
<td>38</td>
<td>32014</td>
<td>25943</td>
<td>47095</td>
<td>45648</td>
<td>51650</td>
<td>86289</td>
<td>106958</td>
<td>108403</td>
<td>115044</td>
</tr>
<tr>
<td>39</td>
<td>40106</td>
<td>51376</td>
<td>73885</td>
<td>115182</td>
<td>82062</td>
<td>87832</td>
<td>68768</td>
<td>122960</td>
<td>145265</td>
</tr>
<tr>
<td>40</td>
<td>22236</td>
<td>37025</td>
<td>27867</td>
<td>89695</td>
<td>59682</td>
<td>50679</td>
<td>90025</td>
<td>87677</td>
<td>149459</td>
</tr>
</tbody>
</table>
## Appendix C-6

### LS2 Runtime Table

<table>
<thead>
<tr>
<th>LS2 time</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.8</td>
<td>0.98</td>
<td>1.89</td>
<td>2.54</td>
<td>4.51</td>
<td>6.54</td>
<td>9.1</td>
<td>10.98</td>
<td>13.99</td>
</tr>
<tr>
<td>2</td>
<td>0.44</td>
<td>0.97</td>
<td>1.92</td>
<td>2.89</td>
<td>4.2</td>
<td>6.72</td>
<td>8.3</td>
<td>11.09</td>
<td>15.96</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>1.11</td>
<td>1.95</td>
<td>2.92</td>
<td>4.12</td>
<td>6.05</td>
<td>8.08</td>
<td>10.64</td>
<td>15.13</td>
</tr>
<tr>
<td>4</td>
<td>0.45</td>
<td>0.98</td>
<td>1.62</td>
<td>2.76</td>
<td>3.7</td>
<td>7.07</td>
<td>10.01</td>
<td>10.7</td>
<td>15.13</td>
</tr>
<tr>
<td>5</td>
<td>0.47</td>
<td>0.9</td>
<td>2.04</td>
<td>2.73</td>
<td>3.7</td>
<td>6.71</td>
<td>9.91</td>
<td>11.79</td>
<td>14.31</td>
</tr>
<tr>
<td>6</td>
<td>0.44</td>
<td>0.86</td>
<td>1.83</td>
<td>2.92</td>
<td>4.01</td>
<td>5.46</td>
<td>8.52</td>
<td>11.39</td>
<td>14.09</td>
</tr>
<tr>
<td>7</td>
<td>0.42</td>
<td>0.95</td>
<td>1.84</td>
<td>2.71</td>
<td>4.65</td>
<td>6.15</td>
<td>8.74</td>
<td>10.45</td>
<td>14.48</td>
</tr>
<tr>
<td>8</td>
<td>0.48</td>
<td>0.9</td>
<td>1.73</td>
<td>2.89</td>
<td>4.03</td>
<td>6.29</td>
<td>7.88</td>
<td>12.17</td>
<td>15.66</td>
</tr>
<tr>
<td>9</td>
<td>0.56</td>
<td>0.97</td>
<td>1.83</td>
<td>2.71</td>
<td>4.62</td>
<td>6.99</td>
<td>7.6</td>
<td>11.97</td>
<td>14.88</td>
</tr>
<tr>
<td>10</td>
<td>0.47</td>
<td>0.86</td>
<td>1.9</td>
<td>2.81</td>
<td>4.32</td>
<td>6.46</td>
<td>8.14</td>
<td>11.9</td>
<td>14.32</td>
</tr>
<tr>
<td>11</td>
<td>0.45</td>
<td>0.86</td>
<td>1.58</td>
<td>2.7</td>
<td>4.04</td>
<td>5.73</td>
<td>8.91</td>
<td>12.12</td>
<td>16.43</td>
</tr>
<tr>
<td>12</td>
<td>0.42</td>
<td>0.98</td>
<td>1.86</td>
<td>2.51</td>
<td>4.59</td>
<td>6.86</td>
<td>9.36</td>
<td>11.89</td>
<td>15.26</td>
</tr>
<tr>
<td>13</td>
<td>0.5</td>
<td>0.94</td>
<td>1.65</td>
<td>2.61</td>
<td>4.23</td>
<td>7.08</td>
<td>8.2</td>
<td>11.81</td>
<td>14.52</td>
</tr>
<tr>
<td>14</td>
<td>0.47</td>
<td>0.95</td>
<td>1.83</td>
<td>3.23</td>
<td>4.46</td>
<td>6.07</td>
<td>8.88</td>
<td>11.2</td>
<td>15.69</td>
</tr>
<tr>
<td>15</td>
<td>0.48</td>
<td>1.05</td>
<td>2.01</td>
<td>2.57</td>
<td>3.85</td>
<td>6.71</td>
<td>9.39</td>
<td>11.47</td>
<td>15.85</td>
</tr>
<tr>
<td>16</td>
<td>0.44</td>
<td>0.89</td>
<td>1.62</td>
<td>2.98</td>
<td>4.74</td>
<td>6.27</td>
<td>8.78</td>
<td>12.98</td>
<td>15.2</td>
</tr>
<tr>
<td>17</td>
<td>0.42</td>
<td>0.9</td>
<td>1.83</td>
<td>2.75</td>
<td>4.21</td>
<td>6.82</td>
<td>8.75</td>
<td>11.75</td>
<td>14.53</td>
</tr>
<tr>
<td>18</td>
<td>0.42</td>
<td>0.89</td>
<td>1.68</td>
<td>2.96</td>
<td>3.87</td>
<td>6.01</td>
<td>8.14</td>
<td>12.5</td>
<td>14.62</td>
</tr>
<tr>
<td>19</td>
<td>0.5</td>
<td>0.83</td>
<td>1.98</td>
<td>2.56</td>
<td>4.1</td>
<td>5.93</td>
<td>8.5</td>
<td>11</td>
<td>14.29</td>
</tr>
<tr>
<td>20</td>
<td>0.44</td>
<td>0.97</td>
<td>1.84</td>
<td>2.82</td>
<td>4.49</td>
<td>6.89</td>
<td>9.22</td>
<td>11.51</td>
<td>15.93</td>
</tr>
<tr>
<td>21</td>
<td>0.47</td>
<td>1.03</td>
<td>1.9</td>
<td>2.79</td>
<td>4.62</td>
<td>7.22</td>
<td>9.76</td>
<td>10.09</td>
<td>15.3</td>
</tr>
<tr>
<td>22</td>
<td>0.47</td>
<td>1.04</td>
<td>1.78</td>
<td>2.68</td>
<td>4.73</td>
<td>6.38</td>
<td>8.91</td>
<td>12.42</td>
<td>13.1</td>
</tr>
<tr>
<td>23</td>
<td>0.41</td>
<td>0.95</td>
<td>1.78</td>
<td>3.15</td>
<td>3.87</td>
<td>6.68</td>
<td>8.41</td>
<td>10.89</td>
<td>14.73</td>
</tr>
<tr>
<td>24</td>
<td>0.42</td>
<td>0.9</td>
<td>2.07</td>
<td>2.95</td>
<td>3.56</td>
<td>6.72</td>
<td>8.24</td>
<td>11.45</td>
<td>14.54</td>
</tr>
<tr>
<td>25</td>
<td>0.53</td>
<td>1</td>
<td>1.82</td>
<td>2.61</td>
<td>4.46</td>
<td>6.58</td>
<td>8.36</td>
<td>12.06</td>
<td>14.46</td>
</tr>
<tr>
<td>26</td>
<td>0.42</td>
<td>1.08</td>
<td>1.7</td>
<td>2.5</td>
<td>4.04</td>
<td>5.68</td>
<td>8.38</td>
<td>10.06</td>
<td>15.32</td>
</tr>
<tr>
<td>27</td>
<td>0.47</td>
<td>0.98</td>
<td>2.01</td>
<td>2.89</td>
<td>4.32</td>
<td>5.99</td>
<td>8.28</td>
<td>11.17</td>
<td>13.26</td>
</tr>
<tr>
<td>28</td>
<td>0.45</td>
<td>0.97</td>
<td>1.84</td>
<td>2.56</td>
<td>4.21</td>
<td>7.27</td>
<td>8.53</td>
<td>10.36</td>
<td>15.96</td>
</tr>
<tr>
<td>29</td>
<td>0.48</td>
<td>1</td>
<td>1.95</td>
<td>2.93</td>
<td>4.07</td>
<td>6.24</td>
<td>9.61</td>
<td>11.75</td>
<td>14.01</td>
</tr>
<tr>
<td>30</td>
<td>0.44</td>
<td>0.97</td>
<td>1.73</td>
<td>2.78</td>
<td>3.87</td>
<td>6.58</td>
<td>8.19</td>
<td>10.58</td>
<td>15.94</td>
</tr>
<tr>
<td>31</td>
<td>0.47</td>
<td>1.06</td>
<td>1.79</td>
<td>2.82</td>
<td>4.26</td>
<td>6.83</td>
<td>8.11</td>
<td>11.75</td>
<td>14.94</td>
</tr>
<tr>
<td>32</td>
<td>0.45</td>
<td>1.03</td>
<td>1.59</td>
<td>3.07</td>
<td>4.15</td>
<td>6.19</td>
<td>7.74</td>
<td>11.51</td>
<td>15.83</td>
</tr>
<tr>
<td>33</td>
<td>0.45</td>
<td>1</td>
<td>1.75</td>
<td>2.73</td>
<td>4.31</td>
<td>6.15</td>
<td>8.25</td>
<td>10.86</td>
<td>13.79</td>
</tr>
<tr>
<td>34</td>
<td>0.44</td>
<td>0.89</td>
<td>1.81</td>
<td>2.73</td>
<td>4.65</td>
<td>6.96</td>
<td>8.14</td>
<td>11.5</td>
<td>14.26</td>
</tr>
<tr>
<td>35</td>
<td>0.42</td>
<td>0.92</td>
<td>1.69</td>
<td>2.76</td>
<td>4.52</td>
<td>6.07</td>
<td>8.86</td>
<td>9.95</td>
<td>14.47</td>
</tr>
<tr>
<td>36</td>
<td>0.37</td>
<td>0.95</td>
<td>1.92</td>
<td>2.75</td>
<td>4.24</td>
<td>6.16</td>
<td>8.24</td>
<td>11.67</td>
<td>17.53</td>
</tr>
<tr>
<td>37</td>
<td>0.42</td>
<td>0.92</td>
<td>1.76</td>
<td>2.62</td>
<td>3.46</td>
<td>5.74</td>
<td>8.33</td>
<td>11.3</td>
<td>14.68</td>
</tr>
<tr>
<td>38</td>
<td>0.42</td>
<td>0.92</td>
<td>1.54</td>
<td>2.78</td>
<td>4.12</td>
<td>6.47</td>
<td>8.52</td>
<td>11.7</td>
<td>14.63</td>
</tr>
<tr>
<td>39</td>
<td>0.47</td>
<td>1.01</td>
<td>1.86</td>
<td>3.48</td>
<td>4.27</td>
<td>6.15</td>
<td>7.88</td>
<td>11.54</td>
<td>15.71</td>
</tr>
<tr>
<td>40</td>
<td>0.44</td>
<td>1</td>
<td>1.59</td>
<td>3.28</td>
<td>3.98</td>
<td>5.21</td>
<td>8.11</td>
<td>11.04</td>
<td>15.98</td>
</tr>
</tbody>
</table>

78
## Appendix C-7

### LS3 Objective Value Table

<table>
<thead>
<tr>
<th>LS3 obj</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4538</td>
<td>4817</td>
<td>6433</td>
<td>4018</td>
<td>10333</td>
<td>11349</td>
<td>10265</td>
<td>10115</td>
<td>15161</td>
</tr>
<tr>
<td>2</td>
<td>2761</td>
<td>6536</td>
<td>7708</td>
<td>6129</td>
<td>7224</td>
<td>10506</td>
<td>9722</td>
<td>12170</td>
<td>17389</td>
</tr>
<tr>
<td>3</td>
<td>2034</td>
<td>5209</td>
<td>7627</td>
<td>8691</td>
<td>5996</td>
<td>9270</td>
<td>7518</td>
<td>8685</td>
<td>11851</td>
</tr>
<tr>
<td>4</td>
<td>3202</td>
<td>4829</td>
<td>4862</td>
<td>7712</td>
<td>3884</td>
<td>11000</td>
<td>13756</td>
<td>8331</td>
<td>13499</td>
</tr>
<tr>
<td>5</td>
<td>2954</td>
<td>2688</td>
<td>5354</td>
<td>7068</td>
<td>5459</td>
<td>12316</td>
<td>14596</td>
<td>9668</td>
<td>11450</td>
</tr>
<tr>
<td>6</td>
<td>2582</td>
<td>4177</td>
<td>5676</td>
<td>8227</td>
<td>4891</td>
<td>6417</td>
<td>12110</td>
<td>12306</td>
<td>10193</td>
</tr>
<tr>
<td>7</td>
<td>2373</td>
<td>4671</td>
<td>6143</td>
<td>6150</td>
<td>10113</td>
<td>7935</td>
<td>11203</td>
<td>7950</td>
<td>14581</td>
</tr>
<tr>
<td>8</td>
<td>2765</td>
<td>4285</td>
<td>4274</td>
<td>6288</td>
<td>7769</td>
<td>8775</td>
<td>6994</td>
<td>12553</td>
<td>15166</td>
</tr>
<tr>
<td>9</td>
<td>3714</td>
<td>3130</td>
<td>5612</td>
<td>6569</td>
<td>9153</td>
<td>12633</td>
<td>6125</td>
<td>11803</td>
<td>12952</td>
</tr>
<tr>
<td>10</td>
<td>3619</td>
<td>2809</td>
<td>6816</td>
<td>4387</td>
<td>8422</td>
<td>9817</td>
<td>8657</td>
<td>13129</td>
<td>13820</td>
</tr>
<tr>
<td>11</td>
<td>2619</td>
<td>3787</td>
<td>4330</td>
<td>4249</td>
<td>8087</td>
<td>5721</td>
<td>10959</td>
<td>15827</td>
<td>19127</td>
</tr>
<tr>
<td>12</td>
<td>2646</td>
<td>5004</td>
<td>7279</td>
<td>6114</td>
<td>10137</td>
<td>10228</td>
<td>11666</td>
<td>15103</td>
<td>16445</td>
</tr>
<tr>
<td>13</td>
<td>3582</td>
<td>4470</td>
<td>3802</td>
<td>6425</td>
<td>7542</td>
<td>8870</td>
<td>9048</td>
<td>10723</td>
<td>9305</td>
</tr>
<tr>
<td>14</td>
<td>2980</td>
<td>3959</td>
<td>6904</td>
<td>9366</td>
<td>8056</td>
<td>11051</td>
<td>10583</td>
<td>12521</td>
<td>13466</td>
</tr>
<tr>
<td>15</td>
<td>4332</td>
<td>4481</td>
<td>6163</td>
<td>4308</td>
<td>8647</td>
<td>11153</td>
<td>12546</td>
<td>9024</td>
<td>14894</td>
</tr>
<tr>
<td>16</td>
<td>2585</td>
<td>2852</td>
<td>5180</td>
<td>7834</td>
<td>8329</td>
<td>10231</td>
<td>8943</td>
<td>14760</td>
<td>16447</td>
</tr>
<tr>
<td>17</td>
<td>2651</td>
<td>3308</td>
<td>6022</td>
<td>5866</td>
<td>7611</td>
<td>10724</td>
<td>12157</td>
<td>15275</td>
<td>14939</td>
</tr>
<tr>
<td>18</td>
<td>4450</td>
<td>3333</td>
<td>3950</td>
<td>5977</td>
<td>5541</td>
<td>9995</td>
<td>6629</td>
<td>13119</td>
<td>14521</td>
</tr>
<tr>
<td>19</td>
<td>3077</td>
<td>3565</td>
<td>6721</td>
<td>5694</td>
<td>4281</td>
<td>9212</td>
<td>10781</td>
<td>11642</td>
<td>12749</td>
</tr>
<tr>
<td>20</td>
<td>1202</td>
<td>4430</td>
<td>7026</td>
<td>7268</td>
<td>9633</td>
<td>10838</td>
<td>14132</td>
<td>8277</td>
<td>15572</td>
</tr>
<tr>
<td>21</td>
<td>31080</td>
<td>31407</td>
<td>60602</td>
<td>53585</td>
<td>97877</td>
<td>97236</td>
<td>133649</td>
<td>58855</td>
<td>131905</td>
</tr>
<tr>
<td>22</td>
<td>20348</td>
<td>54301</td>
<td>47785</td>
<td>52229</td>
<td>68730</td>
<td>72554</td>
<td>76727</td>
<td>159073</td>
<td>77490</td>
</tr>
<tr>
<td>23</td>
<td>27833</td>
<td>52491</td>
<td>38472</td>
<td>89922</td>
<td>38225</td>
<td>90849</td>
<td>80927</td>
<td>86878</td>
<td>100276</td>
</tr>
<tr>
<td>24</td>
<td>12056</td>
<td>16468</td>
<td>62625</td>
<td>79115</td>
<td>26949</td>
<td>83000</td>
<td>87894</td>
<td>90368</td>
<td>102669</td>
</tr>
<tr>
<td>25</td>
<td>32111</td>
<td>42705</td>
<td>40971</td>
<td>63320</td>
<td>68972</td>
<td>87170</td>
<td>86803</td>
<td>95579</td>
<td>124632</td>
</tr>
<tr>
<td>26</td>
<td>13764</td>
<td>49106</td>
<td>46455</td>
<td>30216</td>
<td>53523</td>
<td>56824</td>
<td>80017</td>
<td>82388</td>
<td>122953</td>
</tr>
<tr>
<td>27</td>
<td>19092</td>
<td>37512</td>
<td>60117</td>
<td>48530</td>
<td>83721</td>
<td>79541</td>
<td>114203</td>
<td>106627</td>
<td>49926</td>
</tr>
<tr>
<td>28</td>
<td>21192</td>
<td>39746</td>
<td>47520</td>
<td>41820</td>
<td>103376</td>
<td>79853</td>
<td>120143</td>
<td>86086</td>
<td>139021</td>
</tr>
<tr>
<td>29</td>
<td>27095</td>
<td>21237</td>
<td>55303</td>
<td>58728</td>
<td>40267</td>
<td>78391</td>
<td>168707</td>
<td>124995</td>
<td>122938</td>
</tr>
<tr>
<td>30</td>
<td>20211</td>
<td>28316</td>
<td>70986</td>
<td>46018</td>
<td>41129</td>
<td>99680</td>
<td>116440</td>
<td>59367</td>
<td>132600</td>
</tr>
<tr>
<td>31</td>
<td>23077</td>
<td>31775</td>
<td>42356</td>
<td>58042</td>
<td>93126</td>
<td>92033</td>
<td>60053</td>
<td>103177</td>
<td>114588</td>
</tr>
<tr>
<td>32</td>
<td>22330</td>
<td>42033</td>
<td>27731</td>
<td>78830</td>
<td>63173</td>
<td>81380</td>
<td>80768</td>
<td>93010</td>
<td>110124</td>
</tr>
<tr>
<td>33</td>
<td>27406</td>
<td>46770</td>
<td>56153</td>
<td>61387</td>
<td>56897</td>
<td>61713</td>
<td>104296</td>
<td>127494</td>
<td>88521</td>
</tr>
<tr>
<td>34</td>
<td>12778</td>
<td>25444</td>
<td>45431</td>
<td>49526</td>
<td>85725</td>
<td>138916</td>
<td>73954</td>
<td>111672</td>
<td>94790</td>
</tr>
<tr>
<td>35</td>
<td>8967</td>
<td>24988</td>
<td>38846</td>
<td>42605</td>
<td>83446</td>
<td>67822</td>
<td>93841</td>
<td>73345</td>
<td>127671</td>
</tr>
<tr>
<td>36</td>
<td>22413</td>
<td>39313</td>
<td>76824</td>
<td>49720</td>
<td>63991</td>
<td>95574</td>
<td>68346</td>
<td>125623</td>
<td>193593</td>
</tr>
<tr>
<td>37</td>
<td>30501</td>
<td>23988</td>
<td>43153</td>
<td>65073</td>
<td>29137</td>
<td>35621</td>
<td>53560</td>
<td>90781</td>
<td>110353</td>
</tr>
<tr>
<td>38</td>
<td>31738</td>
<td>23139</td>
<td>44905</td>
<td>39427</td>
<td>43733</td>
<td>78414</td>
<td>98074</td>
<td>96167</td>
<td>104259</td>
</tr>
<tr>
<td>39</td>
<td>40099</td>
<td>50181</td>
<td>71324</td>
<td>114266</td>
<td>76779</td>
<td>81708</td>
<td>54446</td>
<td>114131</td>
<td>134183</td>
</tr>
<tr>
<td>40</td>
<td>21628</td>
<td>36055</td>
<td>23372</td>
<td>87353</td>
<td>52692</td>
<td>37259</td>
<td>83547</td>
<td>73796</td>
<td>141810</td>
</tr>
</tbody>
</table>
### Appendix C-8

**LS3 Runtime Table**

<table>
<thead>
<tr>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>60</th>
<th>70</th>
<th>80</th>
<th>90</th>
<th>100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.77</td>
<td>0.97</td>
<td>1.95</td>
<td>2.89</td>
<td>4.71</td>
<td>6.93</td>
<td>9.9</td>
<td>11.47</td>
</tr>
<tr>
<td>2</td>
<td>0.49</td>
<td>1.09</td>
<td>2</td>
<td>2.94</td>
<td>4.28</td>
<td>6.67</td>
<td>8.42</td>
<td>11.84</td>
</tr>
<tr>
<td>3</td>
<td>0.44</td>
<td>1.21</td>
<td>1.95</td>
<td>3.21</td>
<td>4.34</td>
<td>6.17</td>
<td>8.99</td>
<td>11.32</td>
</tr>
<tr>
<td>4</td>
<td>0.46</td>
<td>1.12</td>
<td>1.81</td>
<td>2.87</td>
<td>3.54</td>
<td>7.17</td>
<td>10.23</td>
<td>12.15</td>
</tr>
<tr>
<td>5</td>
<td>0.53</td>
<td>0.91</td>
<td>2.1</td>
<td>2.79</td>
<td>4</td>
<td>7.26</td>
<td>10</td>
<td>12.05</td>
</tr>
<tr>
<td>6</td>
<td>0.53</td>
<td>0.87</td>
<td>1.84</td>
<td>3.07</td>
<td>4.04</td>
<td>5.27</td>
<td>9.38</td>
<td>11.99</td>
</tr>
<tr>
<td>7</td>
<td>0.49</td>
<td>1.03</td>
<td>1.83</td>
<td>2.68</td>
<td>4.71</td>
<td>6.38</td>
<td>8.52</td>
<td>10.11</td>
</tr>
<tr>
<td>8</td>
<td>0.47</td>
<td>1.25</td>
<td>1.91</td>
<td>3.06</td>
<td>4.12</td>
<td>6.52</td>
<td>7.95</td>
<td>13.67</td>
</tr>
<tr>
<td>9</td>
<td>0.49</td>
<td>1.05</td>
<td>1.9</td>
<td>2.8</td>
<td>4.91</td>
<td>8.54</td>
<td>7.78</td>
<td>12.93</td>
</tr>
<tr>
<td>10</td>
<td>0.45</td>
<td>0.94</td>
<td>1.87</td>
<td>3.04</td>
<td>4.56</td>
<td>6.32</td>
<td>9.16</td>
<td>12.96</td>
</tr>
<tr>
<td>11</td>
<td>0.5</td>
<td>0.89</td>
<td>1.58</td>
<td>3.03</td>
<td>4.47</td>
<td>5.99</td>
<td>8.75</td>
<td>12.85</td>
</tr>
<tr>
<td>12</td>
<td>0.44</td>
<td>0.97</td>
<td>1.9</td>
<td>2.48</td>
<td>4.81</td>
<td>7.04</td>
<td>10.01</td>
<td>12.28</td>
</tr>
<tr>
<td>13</td>
<td>0.51</td>
<td>1.01</td>
<td>1.69</td>
<td>2.87</td>
<td>4.29</td>
<td>7.86</td>
<td>8.75</td>
<td>12.75</td>
</tr>
<tr>
<td>14</td>
<td>0.46</td>
<td>0.98</td>
<td>1.82</td>
<td>3.52</td>
<td>4.65</td>
<td>6.14</td>
<td>9.48</td>
<td>11.9</td>
</tr>
<tr>
<td>15</td>
<td>0.48</td>
<td>1.02</td>
<td>2.05</td>
<td>2.73</td>
<td>4.1</td>
<td>6.83</td>
<td>10.09</td>
<td>11.65</td>
</tr>
<tr>
<td>16</td>
<td>0.6</td>
<td>0.92</td>
<td>1.83</td>
<td>2.88</td>
<td>4.92</td>
<td>6.88</td>
<td>9.51</td>
<td>13.79</td>
</tr>
<tr>
<td>17</td>
<td>0.5</td>
<td>0.88</td>
<td>2.01</td>
<td>2.83</td>
<td>4.8</td>
<td>7.04</td>
<td>9.42</td>
<td>12.1</td>
</tr>
<tr>
<td>18</td>
<td>0.44</td>
<td>0.9</td>
<td>1.87</td>
<td>3.17</td>
<td>3.89</td>
<td>6.44</td>
<td>8.96</td>
<td>13.91</td>
</tr>
<tr>
<td>19</td>
<td>0.58</td>
<td>0.83</td>
<td>2.07</td>
<td>2.79</td>
<td>4.21</td>
<td>6.35</td>
<td>9.34</td>
<td>13.09</td>
</tr>
<tr>
<td>20</td>
<td>0.46</td>
<td>1.01</td>
<td>1.82</td>
<td>2.99</td>
<td>4.59</td>
<td>6.54</td>
<td>10.61</td>
<td>13.08</td>
</tr>
<tr>
<td>21</td>
<td>0.5</td>
<td>1.07</td>
<td>1.9</td>
<td>2.91</td>
<td>4.53</td>
<td>7.66</td>
<td>9.52</td>
<td>10.24</td>
</tr>
<tr>
<td>22</td>
<td>0.44</td>
<td>1.09</td>
<td>1.8</td>
<td>2.54</td>
<td>5.28</td>
<td>6.41</td>
<td>9.68</td>
<td>13.66</td>
</tr>
<tr>
<td>23</td>
<td>0.41</td>
<td>0.97</td>
<td>1.87</td>
<td>3.28</td>
<td>3.78</td>
<td>7.17</td>
<td>9.19</td>
<td>10.85</td>
</tr>
<tr>
<td>24</td>
<td>0.4</td>
<td>1.02</td>
<td>2.07</td>
<td>3.15</td>
<td>3.91</td>
<td>6.44</td>
<td>9.24</td>
<td>11.19</td>
</tr>
<tr>
<td>25</td>
<td>0.55</td>
<td>0.99</td>
<td>1.99</td>
<td>2.76</td>
<td>4.62</td>
<td>6.96</td>
<td>8.71</td>
<td>13.14</td>
</tr>
<tr>
<td>26</td>
<td>0.46</td>
<td>1.09</td>
<td>1.97</td>
<td>2.23</td>
<td>4.54</td>
<td>5.85</td>
<td>8.52</td>
<td>10.96</td>
</tr>
<tr>
<td>27</td>
<td>0.47</td>
<td>1.27</td>
<td>1.99</td>
<td>3.07</td>
<td>4.49</td>
<td>6.79</td>
<td>9.53</td>
<td>11.34</td>
</tr>
<tr>
<td>28</td>
<td>0.48</td>
<td>0.93</td>
<td>1.83</td>
<td>2.85</td>
<td>4.69</td>
<td>6.66</td>
<td>8.99</td>
<td>11.14</td>
</tr>
<tr>
<td>29</td>
<td>0.5</td>
<td>1.01</td>
<td>2.74</td>
<td>3</td>
<td>3.88</td>
<td>6.55</td>
<td>10.27</td>
<td>12.57</td>
</tr>
<tr>
<td>30</td>
<td>0.44</td>
<td>0.95</td>
<td>1.67</td>
<td>2.73</td>
<td>3.6</td>
<td>6.54</td>
<td>9.19</td>
<td>10.34</td>
</tr>
<tr>
<td>31</td>
<td>0.45</td>
<td>1.11</td>
<td>2.05</td>
<td>2.83</td>
<td>4.31</td>
<td>6.98</td>
<td>8.08</td>
<td>12.09</td>
</tr>
<tr>
<td>32</td>
<td>0.49</td>
<td>1.05</td>
<td>1.66</td>
<td>2.99</td>
<td>4.32</td>
<td>6.52</td>
<td>8.51</td>
<td>11.74</td>
</tr>
<tr>
<td>33</td>
<td>0.58</td>
<td>1.05</td>
<td>1.83</td>
<td>2.97</td>
<td>4.29</td>
<td>5.92</td>
<td>8.7</td>
<td>11.76</td>
</tr>
<tr>
<td>34</td>
<td>0.42</td>
<td>0.92</td>
<td>1.83</td>
<td>3.01</td>
<td>5.23</td>
<td>7.49</td>
<td>8.81</td>
<td>11.57</td>
</tr>
<tr>
<td>35</td>
<td>0.41</td>
<td>0.88</td>
<td>1.6</td>
<td>2.91</td>
<td>4.75</td>
<td>7.1</td>
<td>8.85</td>
<td>10.5</td>
</tr>
<tr>
<td>36</td>
<td>0.39</td>
<td>1</td>
<td>2.1</td>
<td>2.67</td>
<td>4.3</td>
<td>6.91</td>
<td>8.33</td>
<td>12.99</td>
</tr>
<tr>
<td>37</td>
<td>0.45</td>
<td>1.09</td>
<td>2.03</td>
<td>2.79</td>
<td>3.4</td>
<td>5.55</td>
<td>8.24</td>
<td>12.25</td>
</tr>
<tr>
<td>38</td>
<td>0.45</td>
<td>1.02</td>
<td>1.66</td>
<td>2.94</td>
<td>4.05</td>
<td>6.46</td>
<td>9.08</td>
<td>11.47</td>
</tr>
<tr>
<td>39</td>
<td>0.49</td>
<td>0.97</td>
<td>1.84</td>
<td>3.52</td>
<td>5.1</td>
<td>6.59</td>
<td>8.39</td>
<td>12.13</td>
</tr>
<tr>
<td>40</td>
<td>0.44</td>
<td>1.05</td>
<td>1.54</td>
<td>3.36</td>
<td>3.79</td>
<td>5.31</td>
<td>8.96</td>
<td>11.9</td>
</tr>
</tbody>
</table>

80