DESIGN OF LOOPED WATER DISTRIBUTION NETWORKS USING A HEURISTIC APPROACH

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

HALIL İBRAHIM ATEŞ

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
CIVIL ENGINEERING

AUGUST 2019

Approval of the thesis:

DESIGN OF LOOPED WATER DISTRIBUTION NETWORKS USING A HEURISTIC APPROACH

submitted by **HALIL İBRAHIM ATEŞ** in partial fulfillment of the requirements for the degree of **Master of Science in Civil Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences**

Prof. Dr. Ahmet Türer
Head of Department, **Civil Engineering**

Assoc. Prof. Dr. Nuri Merzi
Supervisor, **Civil Engineering, METU**

Examining Committee Members:

Prof. Dr. Ali Melih Yanmaz
Civil Engineering Dept., METU

Assoc. Prof. Dr. Nuri Merzi
Civil Engineering, METU

Prof. Dr. Zuhal Akyürek
Civil Engineering Dept., METU

Prof. Dr. Elçin Kentel Erdoğan
Civil Engineering Dept., METU

Assoc. Prof. Dr. Yakup Darama
Civil Engineering Dept., Atılım University

Date: 26.08.2019

ABSTRACT

DESIGN OF LOOPED WATER DISTRIBUTION NETWORKS USING A HEURISTIC APPROACH

Ateş, Halil İbrahim
Master of Science, Civil Engineering
Supervisor: Assoc. Prof. Dr. Nuri Merzi

August 2019, 92 pages

This study aims to design pipe diameters of a water distribution network using a heuristic approach algorithm. The heuristic approach algorithm aims to maximize the reliability of the network while minimizing the total cost of the water distribution network, which is a multi-objective optimization problem. In a water distribution network, the resilience index is defined as network reliability (Todini, 2000). The cost of the water distribution network is the total cost of the network pipes. In this heuristic approach, the minimum required pressure limit, water flow velocity upper limit and reliability index are considered. In this study, the heuristic approach algorithm developed by Todini (2000) is coded as a computer software by the integration of hydraulic network solution software (EPANET) calculate engine and C#. A case study is performed in a DMA of N8.3 pressure zone of Ankara network. In this study, the effect of the change in resilience index on Yayla network pipe diameters is surveyed. As a result, it is observed that network reliability can be increased by 7%, increasing the network cost by only 3.5%. The resilience index value of the current Yayla District network is 0.95 and it can be said that existing Yayla District network is overdesigned.

Keywords: Water Distribution Networks, Optimization Methods, Heuristic Approach, Ankara Water Distribution Network

ÖZ

SU DAĞITIM ŞEBEKELERİNİN SEZGİSEL YAKLAŞIMLA TASARIMI

Ateş, Halil İbrahim
Yüksek Lisans, İnşaat Mühendisliği
Tez Danışmanı: Doç. Dr. Nuri Merzi

Ağustos 2019, 92 sayfa

Bu çalışmanın amacı bir sezgisel yaklaşım algoritması kullanarak su dağıtım şebekesi boru çaplarını tasarlamaktır. Sezgisel yaklaşım algoritması, çok amaçlı bir optimizasyon problemi olan su dağıtım şebekesinin toplam maliyetini minimize ederken şebeke güvenilirliğini maksimize etmeyi amaçlar. Su dağıtım şebekesinde şebeke güvenilirliği olarak "resilience index" kullanılmıştır (Todini, 2000). Su dağıtım şebekesinin maliyeti şebeke borularının maliyeti olarak tanımlanmıştır. Sezgisel yaklaşım uygulanırken gerekli minimum basınç limiti, su akış hızı üst limiti ve güvenilirlik indeksi göz önünde bulundurulmuştur. Bu çalışmada Todini (2000) tarafından önerilen sezgisel yaklaşım algoritması hidrolik şebeke çözüm yazılımının (EPANET) hesap motoru ve C# entegrasyonu ile bir bilgisayar programı olarak kodlanmıştır. Geliştirilen bilgisayar programı ile Ankara'nın su dağıtım şebekesinin N8.3 basınç bölgesinde uygulanmıştır. Bu çalışmada resilience index değerindeki değişimin Yayla Mahallesi su dağıtım boru çaplarına olan etkisi araştırılmıştır. Yapılan hesaplamada şebeke maliyetinin 3.5% arttırılmasıyla, şebeke güvenilirliğinin 7% yükseltilebildiği gözlemlenmiştir. Şu andaki Yayla mahallesi şebekesinin resilience index değeri 0.95 bulunmuş olup bu şebekenin aşırı tasarım olduğu söylenebilir.

Anahtar Kelimeler: Su Dağıtım Sistemleri, Optimizasyon Yöntemleri, Sezgisel Yaklaşım, Ankara Su Dağıtım Şebekesi

To my lovely wife “Seda” and my sweetie girl “Meryem”

ACKNOWLEDGEMENTS

I would like to express my deepest respect and gratitude to Assoc. Prof. Dr. Nuri Merzi. I am grateful for his guidance, support and invaluable encouragement to complete this study.

Finally, I thank my family for their support. This study is dedicated to all those who support science.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

DMA:	District Metered Area
WDN:	Water Distribution Network

LIST OF SYMBOLS

P_{tot} : Total energy of a water distribution network

P_{int} : Dissipated power in a water distribution network pipe

P_{ext}: The energy of water delivered to consumers

ɣ : The specific weight of the water

Q_k : Discharge values of k^{th} reservoir

H_k : Pressure head values of k^{th} reservoir

q_i: Discharge values of i^{th} node

$q_{dem,i}$: Water demand at node i

h_i : Pressure head values of i^{th} node

$h_{des,i}$: The minimum required head pressure

$H_{des,i}$: Hydraulic head elevation of i^{th} node

I_r : Resilience index

I_s : Surplus index

I_f : Failure index

I_{fi} : Failure index of i^{th} node

$\mathrm{C}_{\mathrm{ij}}^{\lambda-1}$: Cost of $(\lambda - 1)^{th}$ pipe between i and j points

$\mathrm{C}_{\mathrm{ij}}^{\lambda}$: Cost of λ^{th} pipe between i and j points

$\mathrm{P}_{\mathrm{ij}}^{\lambda-1}$: Power of $(\lambda - 1)^{th}$ pipe between i and j points

$\mathrm{P}_{\mathrm{ij}}^{\lambda}$: Power of λ^{th} pipe between i and j points

CHAPTER 1

INTRODUCTION

Water is the most essential component of life. Throughout history, people have lived close to water resources. Due to the increase in the human population, areas close to water resources became insufficient for living. For this reason, water has started to be transported to where it is needed by water supply systems including water distribution networks (WDN).

WDNs are infrastructure systems that allow water to be collected to sources, transported from sources to storage tanks and distributed water from storage tanks to consumers. The essential objective of WDNs is to ensure that the required pressures and required amount of potable water is delivered to consumers.

Water demand is increasing day by day with the human population. Providing water to the ever increasing human population is one of the main tasks of water utilities in modern cities. The water supply must be provided in a manner that ensures continuous, adequate pressure and sufficient water quality. An overdesigned system caused budget deficit. On the contrary, water utilities may not be able to deliver sufficient pressure and desired quantity of water to customers while aiming to reduce costs. Considering low investment cost and high reliability, all costs and requirements from design to operation of WDNs must be meticulously calculated by water utilities.

WDNs are very costly structures. Many factors affect the optimum design of WDNs; system hydraulics, reliability, diameter of pipes, availability of necessary materials, water quality, population distribution, future population estimation, geographical conditions, number and locations of pipes, pumps, tanks and valves.

In this study, reliability and pipe diameters are considered as design parameters. Required data of elements are given except pipe diameters; reservoir water elevation, layout and length of distribution pipes, nodes' desired demands and topographic elevations are given.

Reliability can be defined as a guarantee for the continuation of the distribution of water to consumers even during local failures of the WDN. In a WDN, cost of the network decreases as the pipe diameters decrease. However, as pipe diameters in the network decrease, head losses increase, and node pressures decrease in the network. Pipe diameters must be large enough to satisfied adequate pressures and desired demands in a WDN. On the other hand, the WDN should be the lowest possible cost. This is a multi-objective optimization problem. The first objective is to maximize reliability of the network, while the second objective is to minimize cost of network.

It is possible to solve a non-linear multi-objective optimization problem using different techniques. Heuristic approach (Todini, 2000) is one of the methods developed to solve the multi-objective problem defined as minimum cost and maximum reliability in a WDN. In optimization problems, heuristic approaches are considered as a quick and practical solution with the near-optimal results. If the problem is too complex to find the optimal solution, heuristic approach does not give optimal results, but the results are satisfactory.

In this thesis, a computer code is written using the heuristic approach algorithm developed by Todini (2000). The software developed in this study will be called as HALO (Heuristic Approach Logic Optimization).

HALO presented in this study is developed by the heuristic approach algorithm proposed by Todini (2000) study. The purpose of developing HALO is to easily implement Todini (2000) heuristic approach to a real WDN in Ankara (N8.3 pressure zone Yayla District).

In Chapter 2 contains literature review. In Chapter 3, components of a WDN is explained. Structure of the heuristic approach are explained in Chapter 4. Then, programming of HALO that is created for this study based on heuristic approach (Todini, 2000) is presented in Chapter 5.

The existing network N8.3 Ankara Yayla DMA is designed by using HALO in Chapter 6. Discussions and recommendation are presented in Chapter 7. Conclusion of the study is presented in Chapter 8.

1.1. The Aim of the Study

In this study, a software called HALO is developed to implement heuristic approach described by Todini (2000) to Yayla District network of N8.3 region in Keçiören, Ankara.

System reliability is defined by resilience index according to (Todini, 2000). Resilience index (I_r) is related to the hydraulic head surplus at nodes rather than the minimum required pressure head under normal loading. This head surplus represents energy storage in critical situations (segment isolations such as pipe failure and repair.) which increases head losses. Head surplus prevents water cut due to pressure reduction. In other words, in case of pressures reduction in WDN, pressure surplus at nodes compensate it to some extent.

WDNs are very costly structures that need to be designed with high reliability. This is a multi-objective non-linear problem. The problem objectives are to maximize reliability and to minimize cost. In this study, multi- objective non-linear optimization problem is solved using by a methodology developed based on the heuristic approach in (Todini, 2000).

The aim of this study is to develop a software called HALO which is developed based on the methodology described by (Todini, 2000). The purpose of developing HALO is to ensure that the method described by (Todini, 2000) can be easily applied to the desired networks. Hence, the methodology developed in (Todini, 2000) is applied by HALO to the Yayla District in Ankara, Keçiören and the results are compared with the current pipe diameters of the Yayla District end of this study.

CHAPTER 2

LITERATURE REVIEW

WDNs are costly structures. Therefore, it is necessary to use an optimization method during network design for decreasing the cost. Researchers have used various methods to design an optimized WDN.

Approaches are divided into two classes to facilitate optimization studies. The first class includes approaches that reduce number of linear programming problems (Alperovits et al, 1977). Instead of evaluating a pipe between adjacent nodes as a single diameter, it splits the pipe into pieces. In the other class, a pipe is thought to have a single diameter value (Todini, 2000).

On the other hand, considering real complex systems in the context of optimization, there are studies that do not require statistical / probabilistically analysis of network performance under various scenarios (Creaco et al., 2016a; Gessler et al., 1985; Prasadet al., 2003; Todini, 2000; Wang et al., 2014).

Studies on WDN optimization can be examined as head-energy and flow-path: pressure head-energy related studies (Gessler et al., 1985; Prasad et al., 2003; Todini, 2000) and flow-path related studies (Tanyimboh et al., 2000; Tanyimboh et al., 2011). Many studies are argued that head-energy based studies yield better results in terms of reliability than flow-path based studies. (Creaco et al., 2014; Greco et al., 2012; Raad et al., 2010).

Cost optimization is not enough when designing a WDN (Todini, 2000). The hydraulic and physical conditions and restrictions of the network must be considered. The surplus index is defined to evaluate the pressure/energy relationship (Gessler et al., 1985). There is a minimum pressure head that is set by water utilities: the pressure at any node in the network cannot be lower than the specified pressure head. Thus, the required pressure is satisfied to all nodes in a WDN (İlbank, 2013).

Surplus Index is the difference between the required pressure (determined by the water authorities) and the lowest pressure head of node in the network. The required pressure is determined as the third story of a building which is equal about 30 meters in Turkey (İlbank, 2013). The surplus index does not give information about the amount of flow delivered to nodes, but only deals with the pressures at the nodes.

Considering the need for an index for an entire network, “resilience index” can be considered (Todini, 2000). According to (Todini, 2000), the resilience index can be defined as the ability to overcome sudden failures in WDNs.

The resilience index described in (Todini, 2000) considers a network as a whole. However, it is observed that more reliable loops are provided when a relationship is established between the diameters of the pipes connected to a node (Prasad et al., 2003). The study of (Creaco et al., 2016a) proves that the modified resilience index that is defined by (Prasad et al., 2003) is a better performance than the resilience index (Todini, 2000). According to (Creaco et al., 2016a), pipe diameters in a loop represent a complementary component for the network reliability.

CHAPTER 3

COMPONENTS OF WATER DISTRIBUTIONS SYSTEMS

3.1. Transmission Lines

Transmission line is the name of the pipes that bring water from the water source to the beginning of the city pipe network. The diameters of the transmission line pipes are the largest in the WDNs as they transmit the whole flow to be delivered to the city.

Technical and economic conditions should be considered when determining the transmission line route. By making the necessary calculations, the flow style of the water is determined; gravity flow, pumped flow or mixed flow. Additional structures (bridges, etc.) should be avoided when determining the route. The transmission line must be short. It should be planned for ease of transportation, operation, and maintenance (İlbank, 2013).

3.2. Pipes

Pipes are elements that carry water from one point to other one in WDNs. Pipes are produced in certain 6 meters of lengths to ease of transportation. There are also elbows in a pipeline. Elbow is used to change the direction of the line.

For modeling purposes, a very long pipeline can be modeled as a single pipe. However, the pipe characteristics should be the same throughout this length (diameter, material, etc.).

The pipe length indicates the entire distance from one node to the other. This length does not have to be straight.

The pipe's nominal diameter indicates the generic name of the pipe, such as 16 inches (400 mm). Due to manufacturing standards, the inner diameter may differ from the nominal diameter. Most pipes' nominal diameters are smaller than the internal diameters.

3.3. Reservoirs

A reservoir is a network element that can supply or receive large volumes of water in a water network, although elevation of the water in the reservoir does not change. Although theoretically, reservoirs can provide an infinite volume of water, there is no endless source in real life. In general, when the network is modeled, it is assumed that the water level of the reservoir does not change.

3.4. Tanks

Unlike reservoirs, storage tanks are structures with limited water volume. They can be completely filled or discharged. In tanks, the water level fluctuates according to water inlet and outlet.

Depending on the shape of the tank, some elevations should be assigned for modeling. Maximum elevation; indicates the maximum level of the water the tank can store. Overflow elevation; means that the tank is overflowing after this level. Similarly, the minimum elevation indicates the minimum water level the tank can supply water to the consumer. The basic elevation is the datum where the other elevations of the tank are measured. There are various tank types: elevation tanks, standpipes, hydropneumatic tanks, ground and buried tanks.

3.5. Pumps

Pumps are vital for WDN. The pumps provide the energy required to transport the water to the service tank and distribute the water within the WDNs. Pumps ensure that the required amount of water is delivered to the WDN with sufficient pressure. Minimum operating pressures in the network are taken up to 20 m in places with a population up to 50,000 and 30 m in larger populations (İlbank, 2013).

3.6. Valves

Valves are WDN elements that can be opened and closed. Thus, their resistance to water can be changed and the movement of the water in the pipe can be controlled. Valves can be divided into five general groups:

- Isolation valves
- Altitude valves
- Air release and vacuum breaking valves
- Control valves
- Directional valves

3.6.1. Isolation Valves

Isolation valves are presumably the most widely used valves in WDNs. Isolation valves can be controlled manually. Thus, they block or release the flow of water. The word "isolation" fully describes the way these valves are used. For example, it may be necessary to shut down part of the system to replace a broken pipe or a leaking joint. With the isolation valves, the water flow is cut off and allows maintenance.

There should be isolation valves in a well-designed WDN. Thus, an emergency or repair cases affect very few users. In some systems, the isolation valve is used to determine the pressure zone limits.

3.6.2. Directional Valves

In systems with directional valves, also called one-way valves, water can flow in one direction. However, it does not allow reverse flow (backflow). If water tries to flow backward, the valve closes. The water flow remains closed until it moves in the appropriate direction.

Pumps have a check valve on the discharge side. This check valve ensures that the reverse flow does not damage the pump. In most hydraulic models, pumps are regarded as a check valve. Check valve does not need to be redefined for calculation models. If a pump does not have a check valve at the discharge side, water will come back when the pump stops.

3.6.3. Altitude Valves

Many water utilities use altitude valves at the point of entry of the pipe into the water tank. The valve closes when the water level of the tank reaches a specific level. This prevents further inflow and prevents overflow. If the flow reverses, the valve opens again. This allows the water to drain from the tank.

Recently developed hydraulic calculation software automatically assigns altitude valves of minimum and maximum tank levels. It does not require a reassignment of minimum and maximum water levels.

However, if the tank does not have a height control valve, tank overflow is possible. This behavior should be included in the model.

3.6.4. Air Release and Air Vacuum Valves

In WDNs, air release valves are used to release trapped air within the system. Trapped air reduces the pressure of the system below the required pressure level. Air release and air vacuum valves are not included in standard water distribution modeling. There are also valve types that perform the functions of both air release and air vacuum valves.

3.6.5. Control Valves

WDNs should provide proper service under a wide range of loading and failure. Control valves are used to manage flows and pressures under many conditions. Control valves can be controlled mechanically or hydraulically. This allows local or remote control of the control valves.

3.6.6. Pressure Reducing Valves (PRVs)

Water authorities use pressure control to reduce the probability of leaks and pipe failures. Control is applied in limited areas. These areas are determined by pressure-reducing valves (PRVs) and similar other limiters.

The use of a single feeding is generally preferred because of its ease of control and monitoring. However, in case of failure, the water supply service is cut. Multi-feed systems increase supply reliability. However, the system becomes complicated and causes instability in the use of PRV.

Pressure Reducing Valves (PRVs) ensure that there is always a convenient and comfortable water usage pressure. These valves provide control of the pressure from the increased cold-water supply to the field requirements.

3.6.7. Flow Control Valves (FCVs)

Pressure control valve; ensure that the pressure caused by impact or any reason in the lifting line does not exceed a predetermined value. It works with the hydraulic energy of the water and does not need any other energy. Opening and closing speed settings can be made with needle valve. It can also be used as a warning valve in some systems. FCVs fully close and provide complete sealing. Additional parts can be installed for variable flow rates.

It can be converted into different functions with additional control elements. Some of those; Pressure reducer, level controls, impact prevention, pump control, and quick relief valve.

3.7. Fire Hydrants

The fire hydrant is called to the system that distributes water and the most important part of fire extinguishing. A fire hydrant that helps spray water all over the area. It is important to intervene externally in the event of uncontrolled fires as a result of the first intervention in the fire hazard. There are two types of underground and surface types.

Besides, fire hydrants are used to prevent sedimentation in the network in areas where the flow rate of water is low. The high flow rate drawn from the fire hydrant is discharged into the sewer system. Thus, high flow water, both the water network is cleaned and the sewage system.

3.8. Junctions

Junctions are meeting points where two or more pipes meet. In some cases, it may also be the endpoint of a pipe (dead-end). Junction points are not actually a component of real water networks. However, it is used as demand points in water networks for modeling of the system.

CHAPTER 4

STRUCTURE OF THE HEURISTIC APPROACH FOR THE DESIGN OF A WATER DISTRIBUTION NETWORK

In this section, the structure of the heuristic approach described by Todini (2000) will be discussed.

4.1. Concept of Resilience index

In a WDN, there are two different costs. The first is the capital cost of the WDN, which is required only at the time of initial construction. The other cost is caused by operation of WDNs. Small diameter pipes should be selected to reduce the initial investment costs of the WDN. However, as the diameter of the pipes in a network decreases, the head loss at nodes increases. Besides, the amount of energy used for pumping also increases. The increase of energy used for pumping increases operating costs. But, since there is no pump in gravity-driven WDNs, no energy is required for operation. There are only initial investment costs in gravity-driven WDNs.

In a WDN where only pipe diameters are considered, a guaranteed way to minimize costs is to form a branched shape WDN. If a WDN is designed as cost-optimized branched shape, the transmission of water can be severely disrupted in case of possible failure. If any pipe isolation occurs in a looped-shape WDN, water can be delivered to more nodes by using alternative routes. Thereby, the WDN can be deliver water to all nodes, even if some pipes are not used. Because of failure case changed water flow velocities and nodal pressures, the network may not be able to provide the pressures and water velocities as before. In other words, this network becomes a different network entirely. As it is seen, it is not enough that a network is designed as a loop to continue to provide water for failure cases.

In a looped WDN, there must be a pressure surplus in addition to required pressure. It means that more energy is needed than required at each node of the WDN. This surplus energy can be defined as the ability to overcome sudden failures in WDNs. This definition also can be used to explain the resilience index in looped networks (Todini, 2000).

A compact form of resilience index can be defined as follows (Creaco et al, 2016a):

$$I_r = \frac{\sum_{k=1}^{n_n} q_{dem,i}(h_i - h_{des,i})}{\sum_{k=1}^{n_r} Q_k H_k \ - \sum_{k=1}^{n_n} q_{dem,i} H_{des,i}} \tag{4.1}$$

The resilience index is the ratio of the power excess supplied to the users to the power excess leaving the sources.

h_i represents head at node i. $h_{des,i}$ is the minimum required head pressure that must be at i^{th} node. The minimum required head pressure is 30 meters in Turkey (İlbank, 2013); $q_{dem,i}$ is water demand at node i. $H_{des,i}$ is the hydraulic head elevation of i^{th} node from the datum.

Surplus index I_s is formulated in (Todini, 2000) as follows:

$$I_s = \min\{h_i - \ h_{des,i}\} \tag{4.2}$$

The surplus index is not used numerically in the optimization process. The surplus index is used to check whether the calculated WDN provides the minimum pressure or not. In other words, the surplus index is the difference between the required minimum head of the node and the minimum head in the WDN. If this difference is less than zero, the WDN cannot meet the required pressure and desired demands. On the other hand, in the case of failure, the pressures in the network decrease. Therefore, surplus head pressure gives the network the ability to tolerate possible pressure reductions.

Failure index I_f is formulated in (Todini, 2000) as follows:

$$I_f = \frac{\sum_{k=1}^{n_n} I_{fi}}{\sum_{k=1}^{n_n} q_{dem,i}\, h_{des,i}} \tag{4.3}$$

Where

$$I_{fi} = f(x) = \begin{cases} 0, & \forall \mathrm{i}:\ h_i \geq h_{des,i} \\ q_{dem,i}\,, & \forall \mathrm{i}:\ h_i \leq h_{des,i} \end{cases}$$

According to (Todini, 2000), failure index can be used to identify infeasibility and compare the effect of pipe failures to the system.

Three indices described above: resilience index, surplus index, and failure index are used in (Todini, 2000) during the heuristic approach. In the next section, the usage of these indices is shown during the heuristic approach.

4.2. Heuristic Approach for the Design of WDNs

Heuristic algorithms are capable of convergence. Heuristic approach algorithms cannot guarantee the exact solution for a problem, but the result is satisfactory (Akyol et al., 2012).

Systems supplying water to cities are designed according to the configuration of cities. Generally, the network layout is predetermined by roads, buildings, industrial zones etc. Therefore, the optimal design has some preliminary constraints, such as demands at nodes, the layout of the network, and the minimum required pressures.

Design demands are determined by population, industrial development, water volume required in case of fire. In this study, daily water consumption per capita is accepted as 170 liters.

In this heuristic approach process, all pipes in the WDN are assigned the largest possible diameters. Thus, the reliability of the network is maximized. In other words, the resilience index of the WDN is about 1. The following steps presented by (Todini, 2000) are followed to reduce to find optimal pipe diameters.

i. A resilience index is specified by the user (Figure 4.1 – step 3).

ii. All pipes in the network are assigned the largest commercially possible diameter by HALO.

iii. The network is calculated with a hydraulic network solver (step 4 in Figure 4.1). In this study, the EPANET library (Rossman, 1993) is used to code HALO.

iv. According to the analysis results, if the failure index (4.3) is null, the procedure starts to reduce the pipe diameters; otherwise the pipe diameters should be increased.

v. In an iteration step, the cost/power dissipation ratio is checked to rank the pipes, whose diameter is reduced first (step 6 in Figure 4.1). The cost/power dissipation ratio is calculated using the formula (4.4).

vi. $$\{i^*, j^*\} = \max_{\{i,j\}}\left[-\frac{C_{ij}^{\lambda-1} - C_{ij}^{\lambda}}{P_{ij}^{\lambda-1} - P_{ij}^{\lambda}}\right] \qquad (4.4)$$

$\{i^*, j^*\}$ represents the starting and ending points of a pipe. Pipe diameters are reduced by following the ranking list that is calculated according to formula (4.4).

There are three controls must be checked before the reduction is accepted (step 9 in Figure 4.1). The first one is velocity the water in pipes. The water velocity in the pipes should not be higher than a predetermined value (2m/s in this study). The second control is the resilience index. The resilience index value must be higher than the resilience index value specified by the user. The third control is the failure index. Failure index value must be negative.

When the procedure described above is finished, only one pipe diameter is reduced and evaluated whether the reduction is accepted or not (step 10 in Figure 4.1). The above procedure applies to all pipes. It is a loop that starts from step 13 to step 8 in Figure 4.1.

After step 8 (Figure 4.1), it is said that there is no need to recalculate the WDN hydraulically after pipe reductions in (Todini, 2000). However, within the scope of this thesis, the WDN is recalculated hydraulically at this step. Therefore, the results of HALO and (Todini, 2000) differ, which can be seen numerically in Section 5.5. The steps shown above are repeated iteratively.

As a result of the heuristic approach, a cost vs. resilience relationship graph can be created. This graph shows all possible solutions of WDN. Due to the nature of the heuristic approach, it can be said that the results of heuristic approach do not have a mathematical certainty, but it is an adequate approach to show a reasonable range of solutions. In this thesis, a heuristic approach study is made considering only pipe diameters of WDN.

However, if any component is added like a pump, the total power of the network is increased. So, it has a positive effect on the resilience index. However, it should not be forgotten to add pump costs.

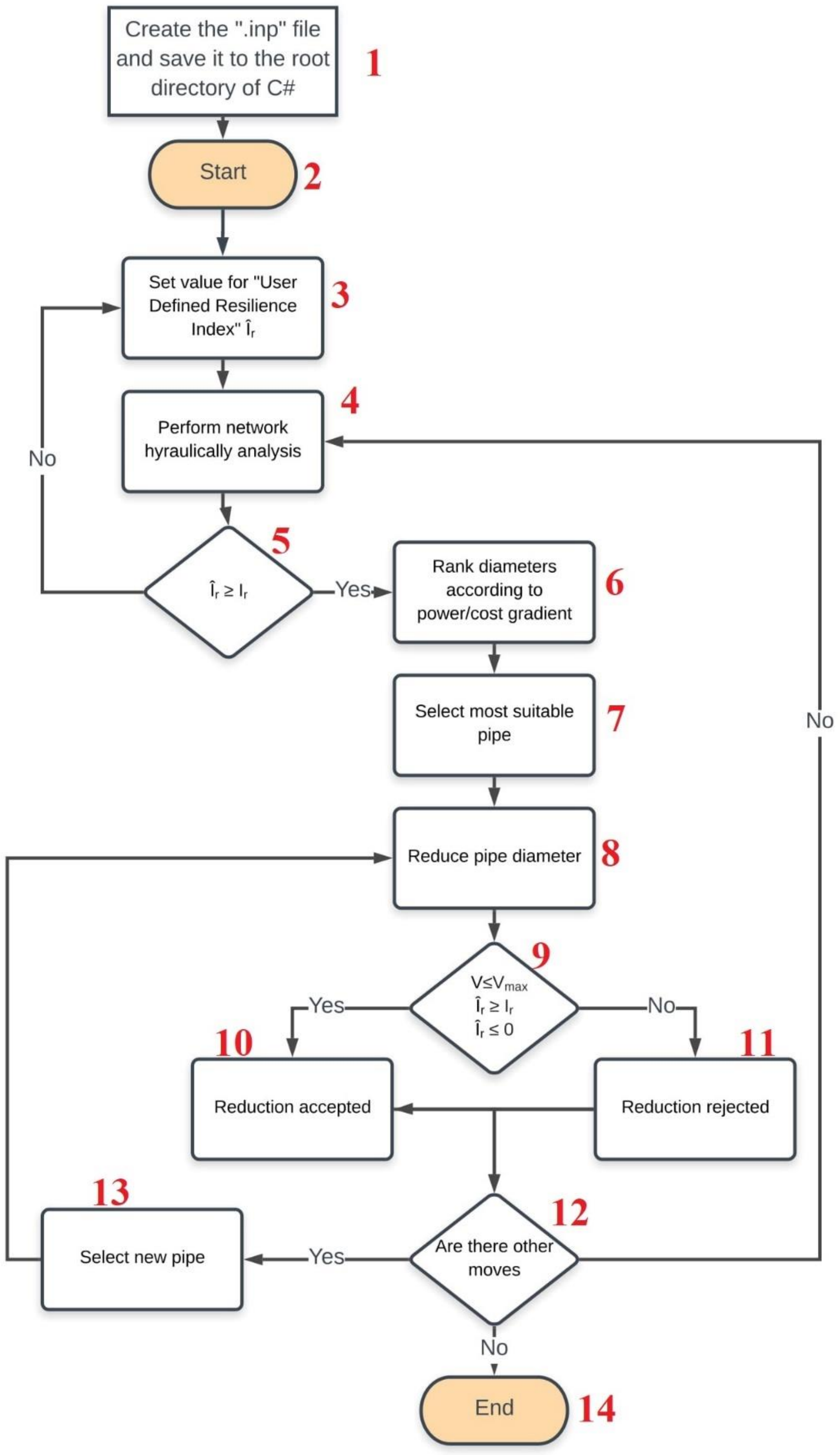

Create the ".inp" file and save it to the root directory of C#
1
Start
2
Set value for "User Defined Resilience Index" $\hat{I}_r$
3
Perform network hyraulically analysis
4
No
$\hat{I}_r \geq I_r$
5
Yes
Rank diameters according to power/cost gradient
6
Select most suitable pipe
7
No
Reduce pipe diameter
8
$V \leq V_{max}$ $\hat{I}_r \geq I_r$ $\hat{I}_r \leq 0$
9
Yes
No
10
Reduction accepted
11
Reduction rejected
13
Select new pipe
Yes
Are there other moves
12
No
End
14

Figure 4.1. Flow chart of the heuristic optimization process (Todini, 2000).

4.3. Pareto Optimal Set

Conflicting objectives increasing difficulty of the multi-objective optimization problems. If there are conflicting objectives in a multi-objective optimization problem, the difficulty of the problem increases further. Trying to maximize one of the objectives while minimizing other objective increases the complexity of the problem. (Kaya et al., 2016).

The Pareto optimal method is one of the solution techniques used in multi-objective optimization problems. Pareto dominance method can be used to select the most appropriate results for multi-purpose optimization problems (Kaya et al., 2016). However, in this study, which solution is preferable is not considered. In this study, a solution cloud graph is obtained that shows only the relationship between the objectives.

It is important to find all possible solutions. This allows the system designer (or decision maker) to evaluate all possible results. The Pareto optimal set of the two-objective optimization problem is shown in Figure 4.2 (Kaya et al., 2016).

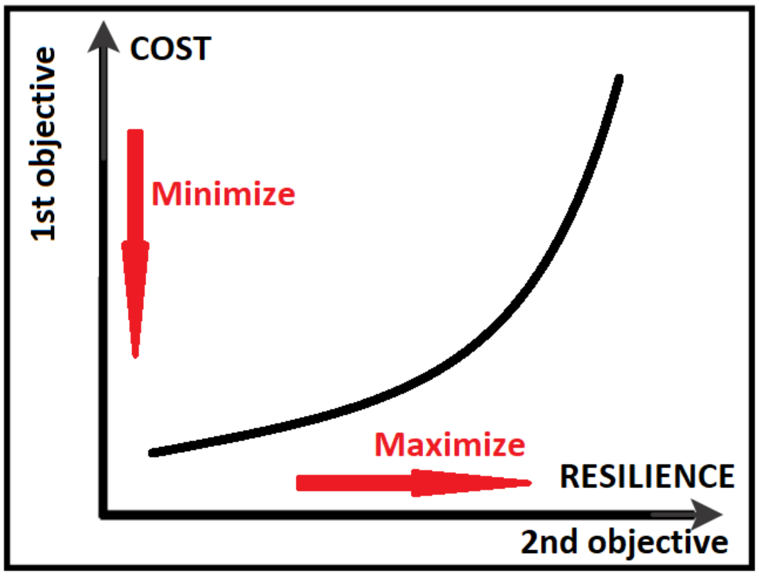


Figure 4.2. Pareto Optimal Set

In this study, there are two objective functions: cost and resilience index. The first objective is to minimize the cost, while the second objective is to maximize the resilience index.

CHAPTER 5

PROGRAMMING THE HEURISTIC OPTIMIZATION PROCESS

The heuristic approach is described in Chapter 4 in detail. It would be the right choice to implement the heuristic approach and related iterations using a computer software.

This section describes the development process, flowchart, and qualification of computer software that applies the heuristic approach method described by (Todini, 2000). In order to apply the heuristic method (Todini, 2000) quickly and accurately, it is necessary to develop computer software in this study.

5.1. Overview of Software Developed for This Study and Operating Logic

In this section, the software developed for this thesis is submitted. HALO is a desktop software developed on the heuristic approach algorithm.

HALO development language is C # (pronounced as C Sharp). Microsoft Visual Studio 2017 is used as a software development environment. The author of this thesis creates HALO by writing approximately three thousand lines of code to implement the heuristic approach of (Todini, 2000). Code of HALO is shared in the Appendices section. Furthermore, the GitHub link is shared in (Web 1).

When describing HALO, it is necessary to mention about the hydraulic network solver calculation library (Web 2). The C # EPANET library is developed on the open-source code of EPANET software developed by (Rossman et al., 2011). It is possible to code EPANET with various programming languages (Web 3). The most important reason for using C# in this study is that C# language has a useful EPANET library.

HALO includes the EPANET calculation engine and the classes in which the necessary hydraulic calculations are made and can run without additional software. HALO can only run on Microsoft Windows operating systems.

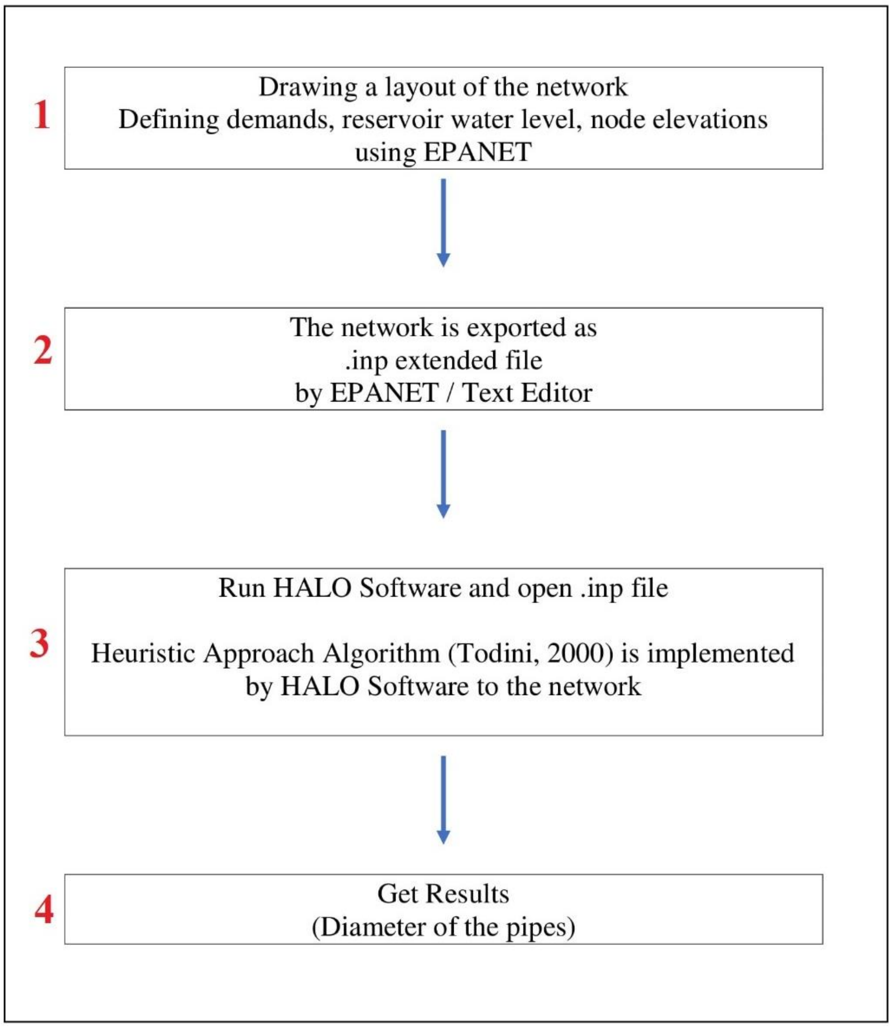

1
Drawing a layout of the network
Defining demands, reservoir water level, node elevations using EPANET
2
The network is exported as .inp extended file by EPANET / Text Editor
3
Run HALO Software and open .inp file
Heuristic Approach Algorithm (Todini, 2000) is implemented by HALO Software to the network
4
Get Results
(Diameter of the pipes)

Figure 5.1. Procedure of HALO

Flowchart of HALO provided in Figure 5.1 has a user-friendly interface (Figure 5.2). In the third step of HALO flowchart in Figure 5.1, the flow chart of the heuristic optimization process (Figure 5.3) developed by (Todini, 2000) is implemented to the network provided by HALO. Figure 5.3 is the flow chart of HALO and EPANET integration according to Todini (2000), however Figure 4.1 represents the procedure of Todini (2000).

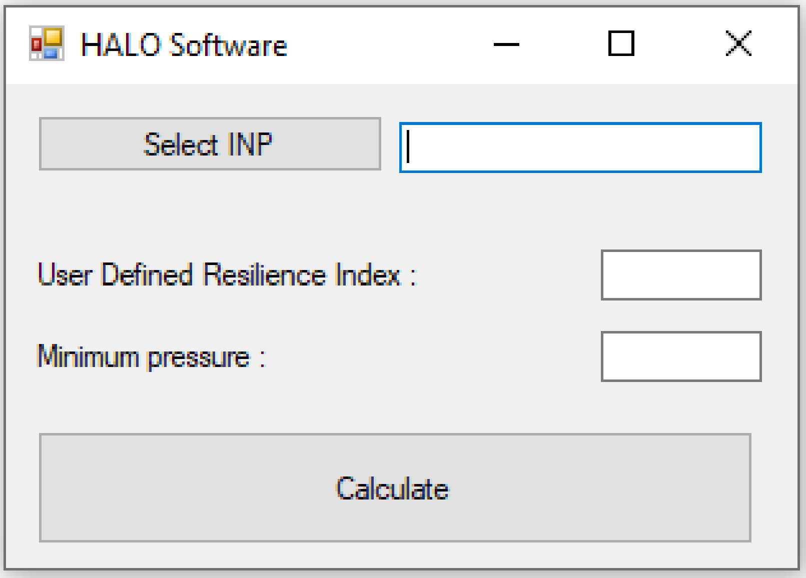


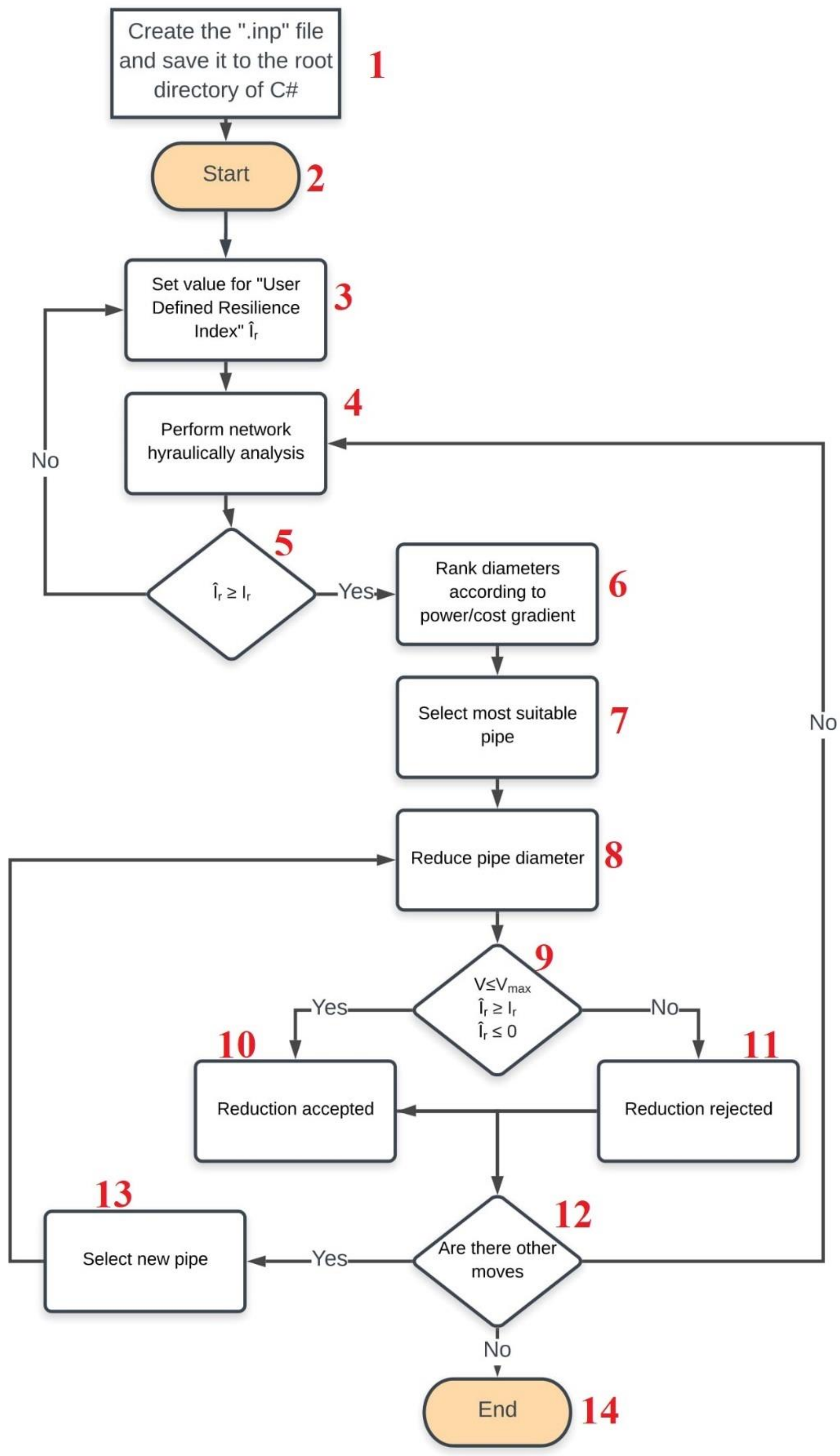
Figure 5.2. User interface of HALO

HALO can only use inp extension files. Inp files that contain all the informations of a WDN are files that can be created by a text editor or hydraulic network solver (Figure 5.4). As shown in Figure 5.4, an inp file contains all the information of a water network, such as reservoir water elevations, node demands and elevations, diameters and lengths of pipes.

The purpose of HALO which is using the method developed by (Todini, 2000) that is described in detail in Chapter 4 is to minimize the costs and to maximize reliability of a WDN. Only pipe diameters are used as variables during the process of HALO. Changes in pipe diameters in a WDN affect both the costs of the system and head losses. If the pipe diameters increase, the head loss in the WDN is reduced. Thus, the network has more surplus heads against pressure reductions in case of failure.

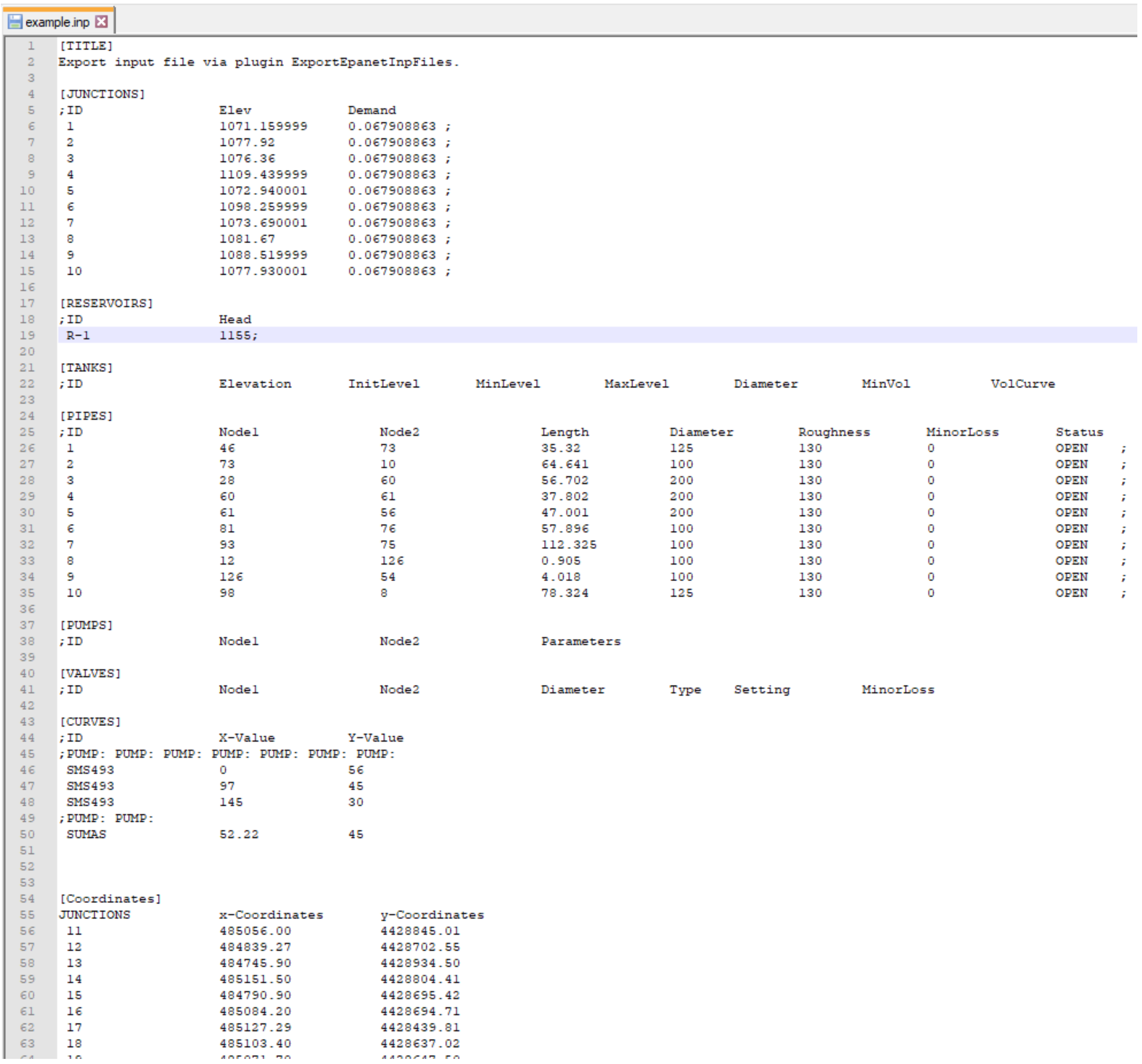
As described above, HALO can work on a network that's coordinates and elevations of demand points, reservoir location and reservoir water level pre-assigned. (Figure 5.1-first step). Due to the heuristic approach, only the pipe diameters in the network are changed. Besides, the EPANET library can perform all hydraulic calculations of a WDN (e.g., water velocity in pipes, pressures at nodes).

The network file whose coordinates and elevations of nodes, reservoir location, reservoir water elevation and nodal demands are assigned by EPANET should be made available to use by HALO. In other words, the WDN should be saved as an inp file (Figure 5.5). Thus, the second step of Figure 5.1 is completed. When the HALO is run, a user-friendly interface opens in which the inp file is first selected (Figure 5.2). The user must define a resilience index. The default value of the minimum required head pressure that can be changed is 30 meters. After the required fields are filled, click the 'Calculate' button. HALO applies the approach defined by (Todini, 2000) to the given network inp file, changes the required pipe diameters, and finally saves the changes in the inp file. A text editor or EPANET can be used to view the final pipe diameters of the given network.



Create the ".inp" file and save it to the root directory of C#
1
Start
2
Set value for "User Defined Resilience Index" $\hat{I}_r$
3
Perform network hyraulically analysis
4
No
$\hat{I}_r \geq I_r$
5
Yes
Rank diameters according to power/cost gradient
6
Select most suitable pipe
7
No
Reduce pipe diameter
8
$V \leq V_{max}$ $\hat{I}_r \geq I_r$ $\hat{I}_r \leq 0$
9
Yes
No
10
Reduction accepted
11
Reduction rejected
13
Select new pipe
Yes
Are there other moves
12
No
End
14

Figure 5.3. Flowchart of HALO and EPANET integration according to Todini (2000)



example.inp

```
[TITLE]
Export input file via plugin ExportEpanetInpFiles.

[JUNCTIONS]
;ID                 Elev            Demand
 1                  1071.159999     0.067908863 ;
 2                  1077.92         0.067908863 ;
 3                  1076.36         0.067908863 ;
 4                  1109.439999     0.067908863 ;
 5                  1072.940001     0.067908863 ;
 6                  1098.259999     0.067908863 ;
 7                  1073.690001     0.067908863 ;
 8                  1081.67         0.067908863 ;
 9                  1088.519999     0.067908863 ;
 10                 1077.930001     0.067908863 ;

[RESERVOIRS]
;ID                 Head
 R-1                1155;

[TANKS]
;ID                 Elevation       InitLevel       MinLevel        MaxLevel        Diameter        MinVol          VolCurve

[PIPES]
;ID                 Node1               Node2               Length          Diameter        Roughness       MinorLoss       Status
 1                  46                  73                  35.32           125             130             0               OPEN    ;
 2                  73                  10                  64.641          100             130             0               OPEN    ;
 3                  28                  60                  56.702          200             130             0               OPEN    ;
 4                  60                  61                  37.802          200             130             0               OPEN    ;
 5                  61                  56                  47.001          200             130             0               OPEN    ;
 6                  81                  76                  57.896          100             130             0               OPEN    ;
 7                  93                  75                  112.325         100             130             0               OPEN    ;
 8                  12                  126                 0.905           100             130             0               OPEN    ;
 9                  126                 54                  4.018           100             130             0               OPEN    ;
 10                 98                  8                   78.324          125             130             0               OPEN    ;

[PUMPS]
;ID                 Node1               Node2               Parameters

[VALVES]
;ID                 Node1               Node2               Diameter        Type    Setting         MinorLoss

[CURVES]
;ID                 X-Value         Y-Value
;PUMP: PUMP: PUMP: PUMP: PUMP: PUMP: PUMP:
 SMS493             0               56
 SMS493             97              45
 SMS493             145             30
;PUMP: PUMP:
 SUMAS              52.22           45



[Coordinates]
JUNCTIONS           x-Coordinates       y-Coordinates
 11                 485056.00           4428845.01
 12                 484839.27           4428702.55
 13                 484745.90           4428934.50
 14                 485151.50           4428804.41
 15                 484790.90           4428695.42
 16                 485084.20           4428694.71
 17                 485127.29           4428439.81
 18                 485103.40           4428637.02
```

Figure 5.4. inp extended file example

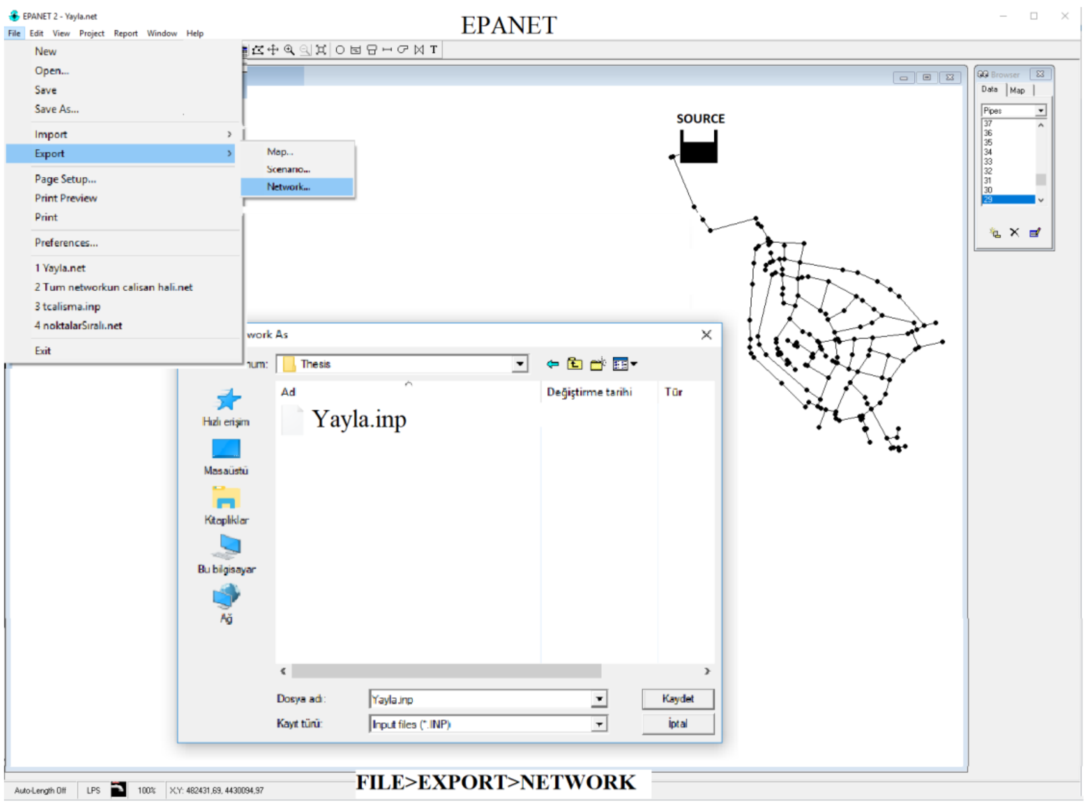


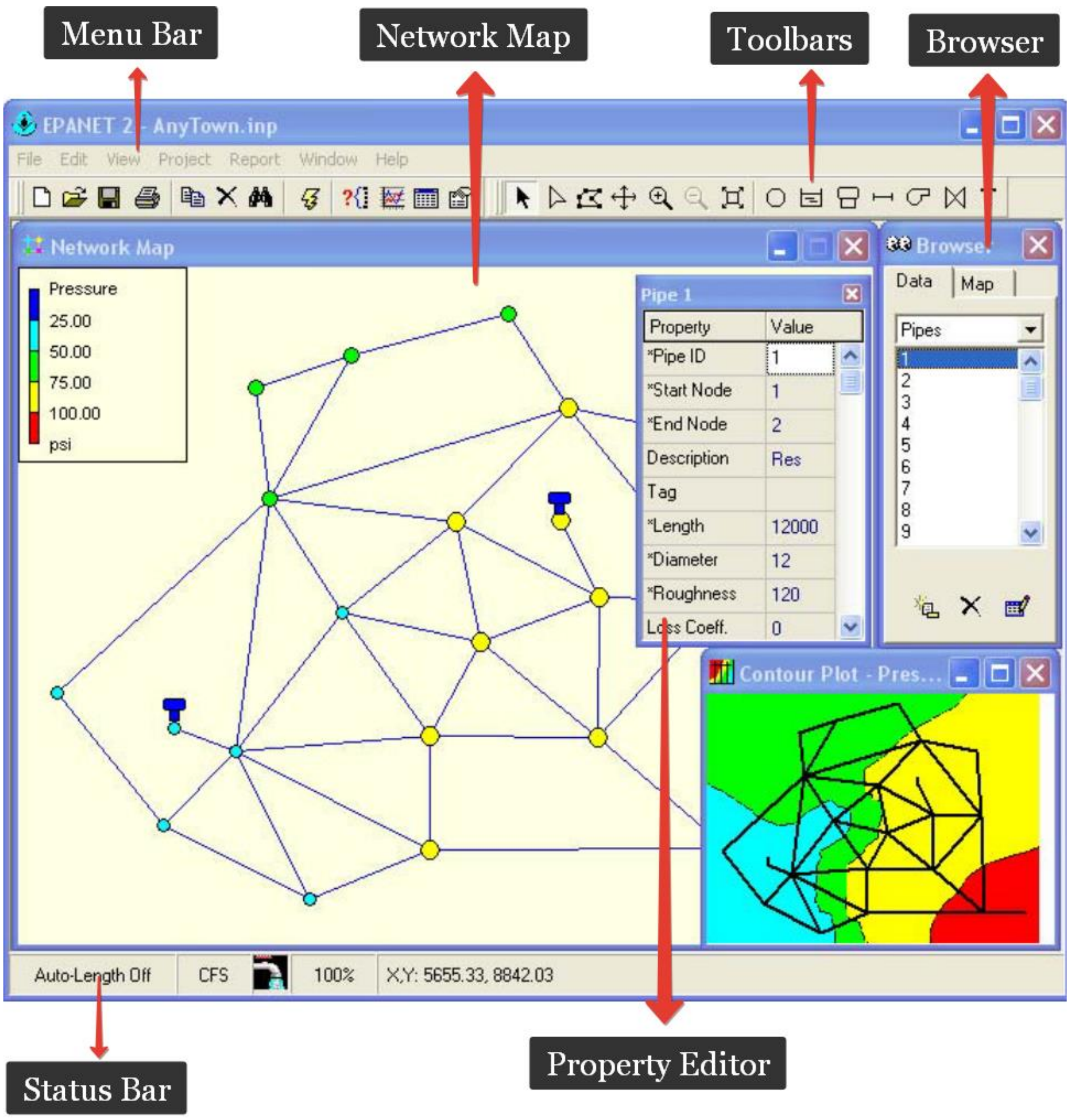
Figure 5.5. Exporting a network file as ".inp" extension format from EPANET

5.2. Hydraulic Network Solver – EPANET

EPANET is software that performs the simulation of hydraulic and water quality behavior in pressurized pipe networks as extended period simulation. In a network, it consists of pipes, pumps, valves and storage tanks or reservoirs. EPANET monitors the flow of water in pipes, pressures at nodes, the water level in tanks and concentration of chemical species (Rossman, 1993).

EPANET is widely used open-source software to simulate the hydraulic and water quality behavior of WDNs over a long period of time (Rossman, 1993). EPANET can be used as a desktop application with a user interface (Figure 5.6) and a Programmer's Toolkit is available to specialize the hydraulic solver for programmers (Rossman et al., 2011).

HALO, which is a software, developed for this study to work properly, an under static loading condition WDN is required. Using the calculation engine of EPANET, HALO can perform the network under static loading.



Pipe 1

Property	Value
*Pipe ID	1
*Start Node	1
*End Node	2
Description	Res
Tag	
*Length	12000
*Diameter	12
*Roughness	120
Loss Coeff.	0

Auto-Length Off | CFS | 100% | X,Y: 5655.33, 8842.03

Figure 5.6. EPANET user interface (Rossman et al., 2011)

5.3. HALO Operation Logic Application on Simple Two-loop Network

The logic of HALO is explained in detail on a simple two-loop network created by (Alperovits et al, 1977). This two-loop simple network is a gravity-driven network under single loading (Figure 5.7).

This network consists of 8 pipes. 14 different pipe diameters from 6 nodes, 1 reservoir, and two loops. The pipe lengths are 1000 meters for each pipe and the Haizen-William coefficient is taken as 130 in all pipes. Demands and elevations in the nodes can be seen in the Table 5.2.

Table 5.1. *Cost of pipes defined in (Todini, 2000)*

Diameters (mm)	Costs ($/m)
25.4	2000
50.8	5000
76.2	8000
101.6	11000
152.4	16000
203.2	23000
254	32000
304.8	50000
355.6	60000
406.4	90000
457.2	130000
508	170000
558.8	300000
609.6	550000

Table 5.2. *Nodal Demands and Elevations for (Alperovits et al, 1977)*

Node	Node Elevations [m]	Node Demands [m3/s]
1 (reservoir)	210	-0.3111
2	150	0.0278
3	160	0.0278
4	155	0.0333
5	150	0.0750
6	165	0.0917
7	160	0.0556

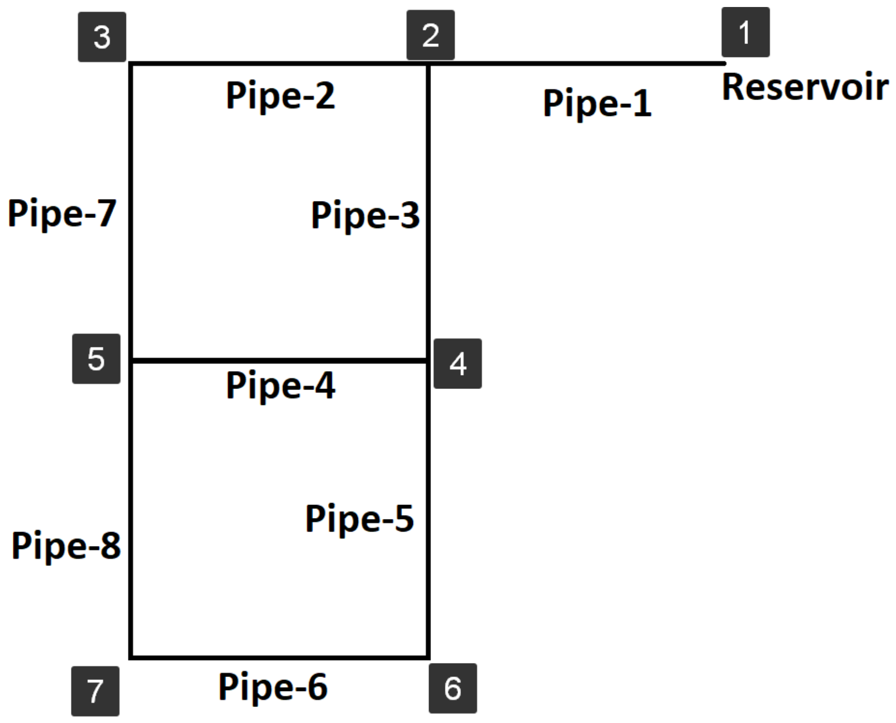


Figure 5.7. Simple two-loop network created by (Alperovits et al, 1977)

5.4. HALO Operation Steps

After clicking the ‘Calculation’ button in HALO interface, the following steps are performed;

1. The user-defined resilience index is considered by HALO. User-defined resilience index indicates the minimum limit value of the resilience index of the WDN.
2. HALO sets the diameter of all pipes in the network as the largest commercially possible diameters (Table 5.1). For this case, all pipe diameters in the network are set to 609.6 mm (Table 5.1).
3. System hydraulic calculation is done by using EPANET (Rossman, 1993) calculation engine.
4. One size smaller diameter is set to pipes (588.5 mm for this case) according to Table 5.1. Then the hydraulical calculation is performed.

5. The head losses and flow velocities in Table 5.3 and Table 5.4 obtained in steps 3 and 4 are calculated as shown in Table 5.5.
6. According to the cost/power dissipation values obtained in step 5, pipes are ordered by descending (Table 5.6).
7. The diameter of the pipe with the largest value in the ranked list (Table 5.6) is reduced first. In this example, pipe-6 is reduced to one step from the list of commercially available pipe diameters (Table 5.1). For this case, Pipe-6 is set a value of 558.8mm.
8. The network is calculated hydraulically.
9. Three checks are made before the reduction is accepted for Pipe-6.
    a. Velocity (It should be less than 2 m/s in this study.)
    b. Resilience index (It must be greater than the resilience index defined by the user in step 1.)
    c. Failure index (It should be negative.)
10. If all the controls in step 9 are correct, the diameter of the Pipe-6 is set to 558.8mm.
11. Step 7, step 8, step 9 and step 10 are applied to all pipes individually. It should be applied to 8 pipes for this case.
12. If at least one of the controls in step 9 is not satisfied, the reduction of the diameter is denied. The pipe whose diameter reduction is rejected is never included in an iteration.
13. The above steps are repeated until there are no more pipes to be reduced in diameter.
14. Last iteration diameters are assigned to the pipes by HALO and saved on the inp file.

Table 5.3. *Pipe diameters are set 609.6mm (first situation)*

First situation				
Label	Diameter (mm)	Flow (m³/h)	Velocity (m/s)	Head loss (m)
Pipe-1	609.6	1,120	1.27	2.54
Pipe-2	609.6	455	0.51	0.48
Pipe-3	609.6	565	0.64	0.72
Pipe-4	609.6	153	0.17	0.06
Pipe-5	609.6	293	0.33	0.21
Pipe-6	609.6	37	0.04	0.004
Pipe-7	609.6	355	0.40	0.30
Pipe-8	609.6	237	0.27	0.14

Table 5.4. *Pipe diameters are set 558.8mm–one size reduced (second situation)*

Second situation				
Label	Diameter (mm)	Flow (m³/h)	Velocity (m/s)	Head loss (m)
Pipe-1	558.8	1,120	1.53	4.04
Pipe-2	558.8	455	0.62	0.76
Pipe-3	558.8	565	0.77	1.14
Pipe-4	558.8	153	0.21	0.10
Pipe-5	558.8	293	0.40	0.34
Pipe-6	558.8	37	0.05	0.01
Pipe-7	558.8	355	0.49	0.48
Pipe-8	558.8	237	0.33	0.23

Table 5.5. *Cost/Power dissipation ratio calculation for Pipe 6*

Pipe 6

	Diameter (mm)	FLOW (m^3/h)	Head loss (m)	COST ($/m)
situation 1	609.6	37	0.004	550 000
situation 2	558.8	37	0.01	300 000

First situation P1= 9.81*(1000/3600) * 37 * (0.004) = 0.403 Watt

Second situation P2= 9.81*(1000/3600) * 37 * (0.01) = 1.008 Watt

First situation Pipe cost=1000 * 550= $550,000.00

Second situation Pipe cost=1000 * 300= $300,000.00

-(300000-550000) / (1.00825-0.403) = **413052.458**

Table 5.6. *Ranked pipes according to Cost/Power dissipation ratio*

Pipe order of diameter reduction	
Pipe-6	**413052.458**
Pipe-4	14991.155
Pipe-8	4301.168
Pipe-5	2408.582
Pipe-7	1435.734
Pipe-2	720.119
Pipe-3	386.612
Pipe-1	54.609

5.5. Result of Two-looped Network Solved by HALO

In this section, the two-loop network created by (Alperovits et al, 1977) is solved by HALO. For this case study (Todini, 2000), solution A with a resilience index of 0.41 is used (Table 5.7 – Second column). The resilience index 0.41 is used in HALO to compare solution of (Todini, 2000).

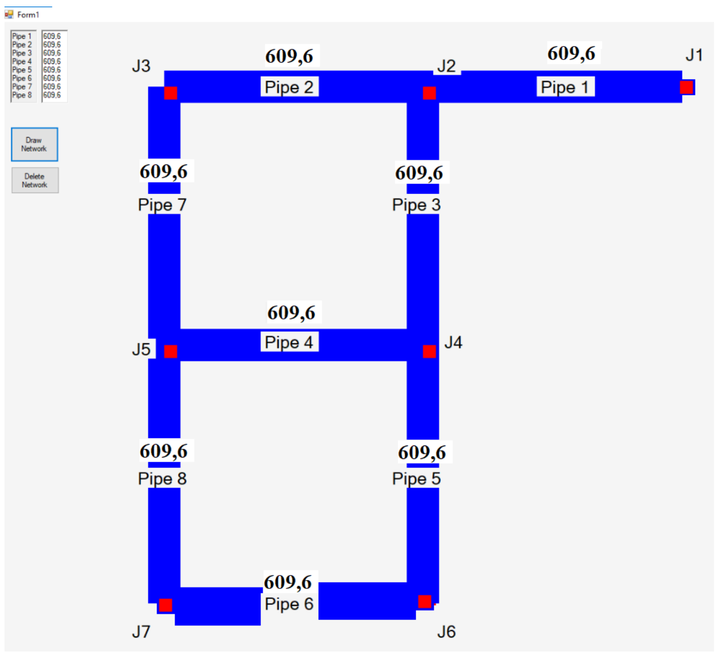
Comparison of (Alperovits et al, 1977) network calculated by (Todini, 2000) vs. HALO can be seen in Table 5.7. As shown in Table 5.7, there are differences between the results of HALO solution and (Todini, 2000). This difference is due to the assumption made by (Todini, 2000) in Step 8 of HALO operating method described in section 5.4. However, in step 8, HALO calculates the network hydraulically. In Table 5.7, it can be clearly seen in the third column that system reliability is not considered in the classical cost optimization method.

Table 5.7. *Comparison of two-loop network result of Todini (2000) vs cost optimization vs HALO*

Pipes	Solution A (Todini, 2000)	Cost Optimization	This Study [HALO]
pipe 1	457.2	457.2	457.2
pipe 2	406.4	254	355.6
pipe 3	355.6	406.4	355.6
pipe 4	152.4	101.6	304.8
pipe 5	355.6	406.4	406.4
pipe 6	25.4	254	25.4
pipe 7	355.6	254	355.6
pipe 8	254	25.4	406.4
Resilience index	**0.41**	**0.22**	**0.411**
Head difference	1.08	0.5	0.57
Cost of the system	$ 450,000	$ 419,000	$ 542,000

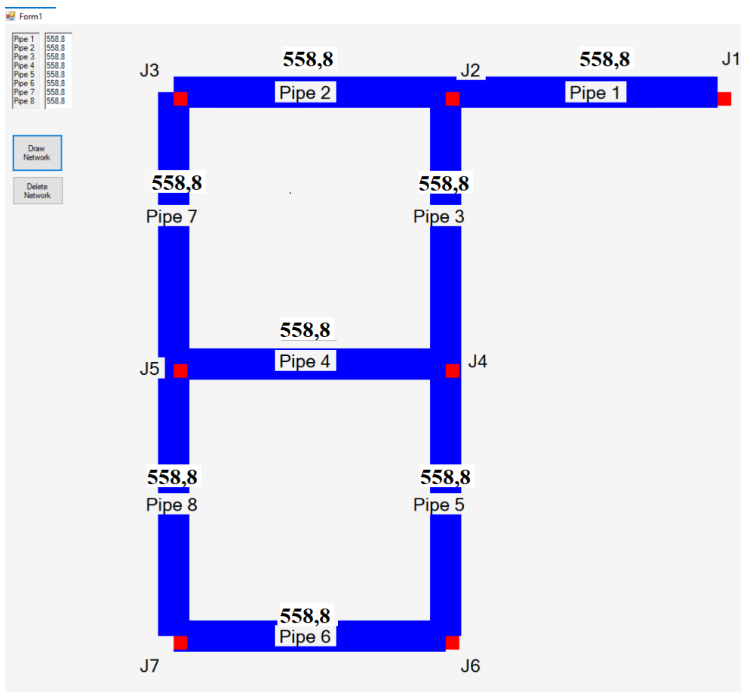
5.6. Visualization of Iteration Steps of Two-loop Network

Pipe diameters in the iteration steps in a two-loop network applied the heuristic approach (Todini, 2000) visualized by HALO.



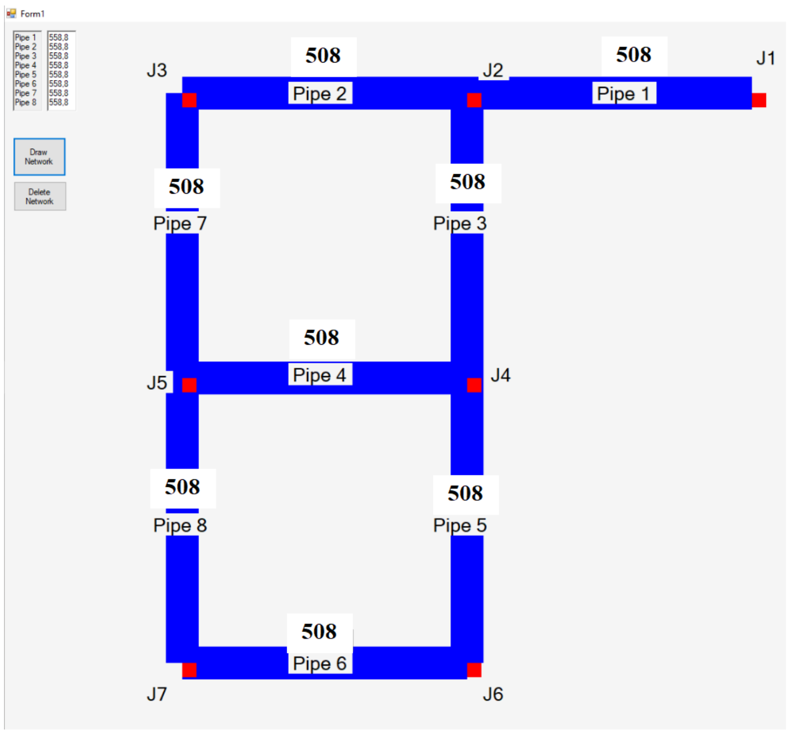
Pipe 1	609,6
Pipe 2	609,6
Pipe 3	609,6
Pipe 4	609,6
Pipe 5	609,6
Pipe 6	609,6
Pipe 7	609,6
Pipe 8	609,6

Figure 5.8. Pipe diameters of two loop network end of the 1st iteration



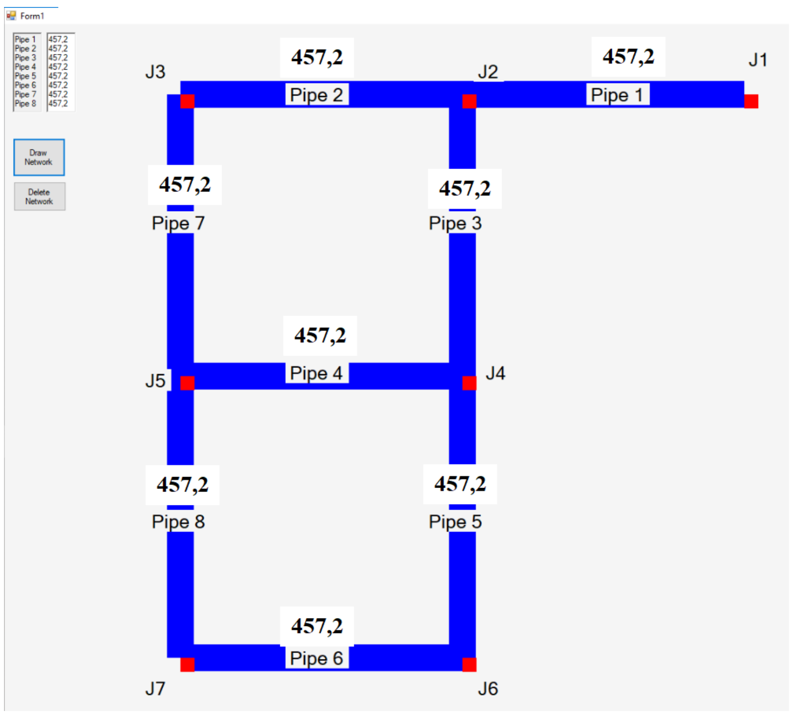
Pipe 1	558,8
Pipe 2	558,8
Pipe 3	558,8
Pipe 4	558,8
Pipe 5	558,8
Pipe 6	558,8
Pipe 7	558,8
Pipe 8	558,8

Figure 5.9. Pipe diameters of two loop network end of the 2nd iteration



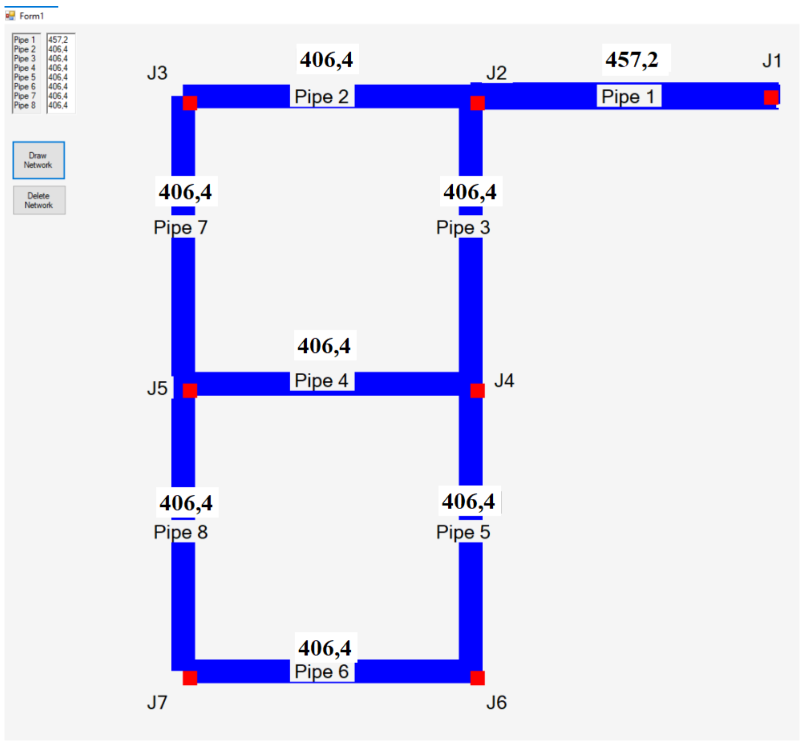
Pipe 1	558,8
Pipe 2	558,8
Pipe 3	558,8
Pipe 4	558,8
Pipe 5	558,8
Pipe 6	558,8
Pipe 7	558,8
Pipe 8	558,8

Figure 5.10. Pipe diameters of two loop network end of the 3rd iteration



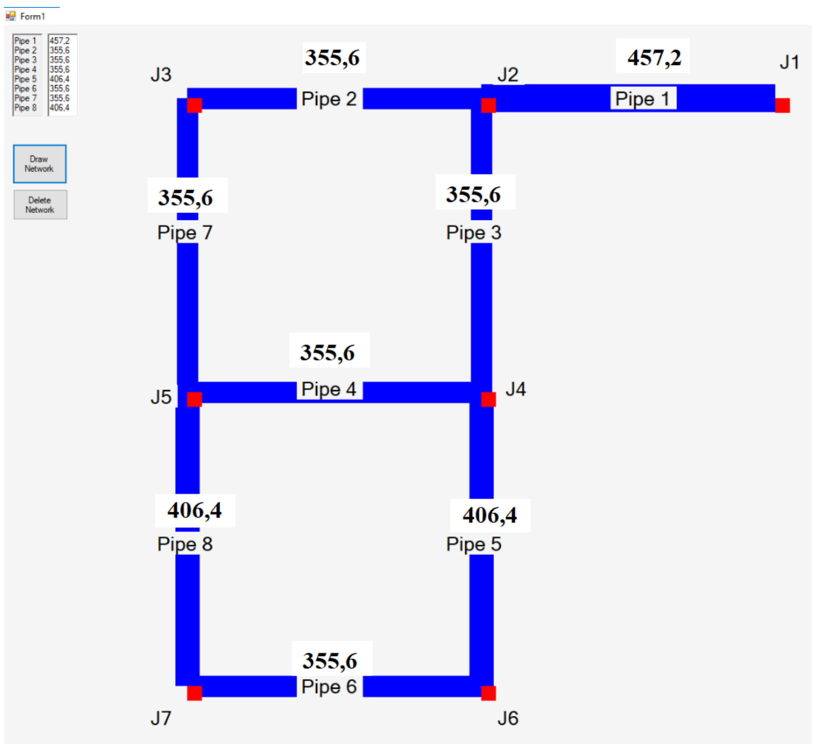
```
Pipe 1  457.2
Pipe 2  457.2
Pipe 3  457.2
Pipe 4  457.2
Pipe 5  457.2
Pipe 6  457.2
Pipe 7  457.2
Pipe 8  457.2
```

Figure 5.11. Pipe diameters of two loop network end of the 4th iteration



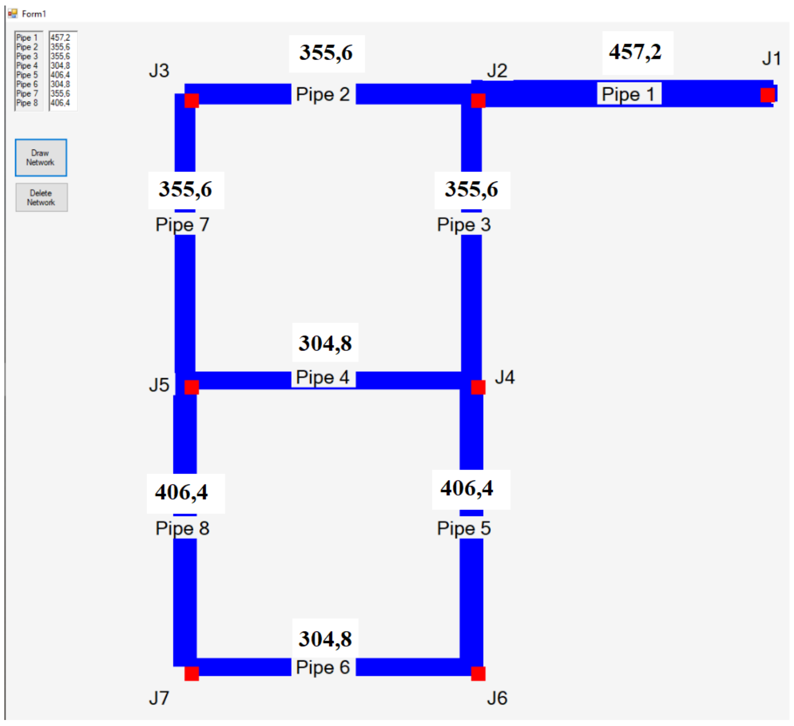
```
Pipe 1  457.2
Pipe 2  406.4
Pipe 3  406.4
Pipe 4  406.4
Pipe 5  406.4
Pipe 6  406.4
Pipe 7  406.4
Pipe 8  406.4
```

Figure 5.12. Pipe diameters of two loop network end of the 5th iteration



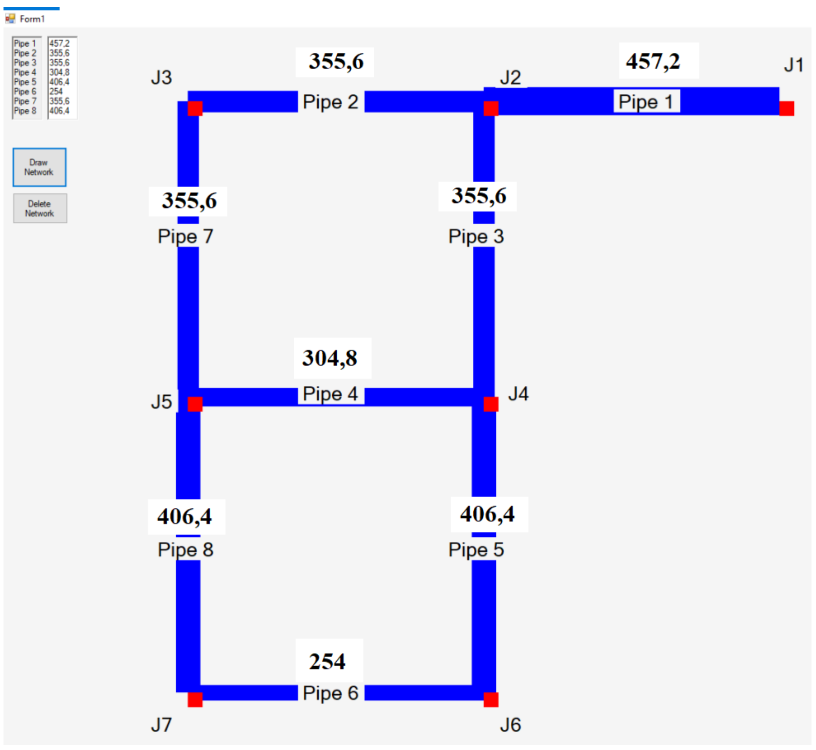
Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	355,6
Pipe 5	406,4
Pipe 6	355,6
Pipe 7	355,6
Pipe 8	406,4

Figure 5.13. Pipe diameters of two loop network end of the 6th iteration



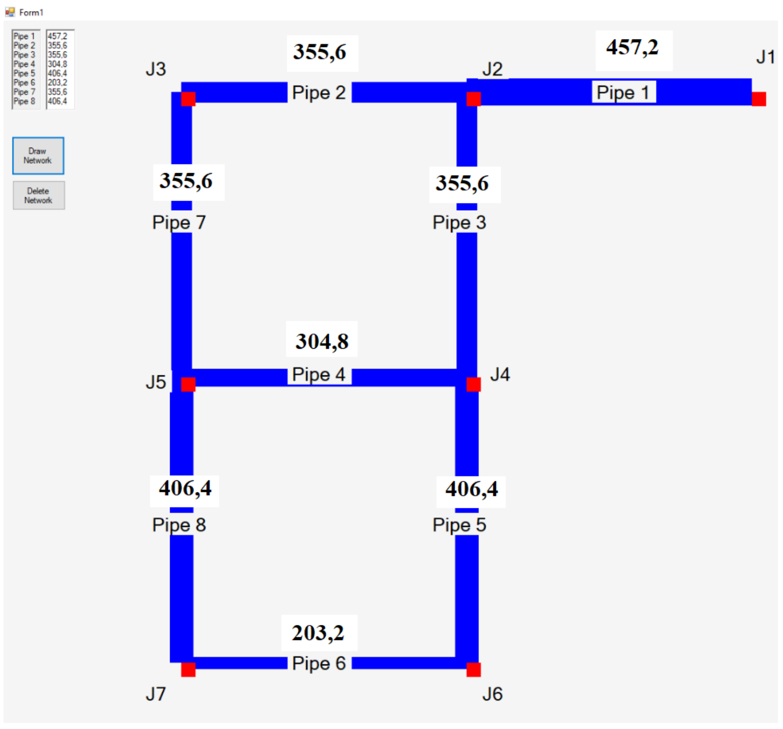
Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	304,8
Pipe 5	406,4
Pipe 6	304,8
Pipe 7	355,6
Pipe 8	406,4

Figure 5.14. Pipe diameters of two loop network end of the 7th iteration



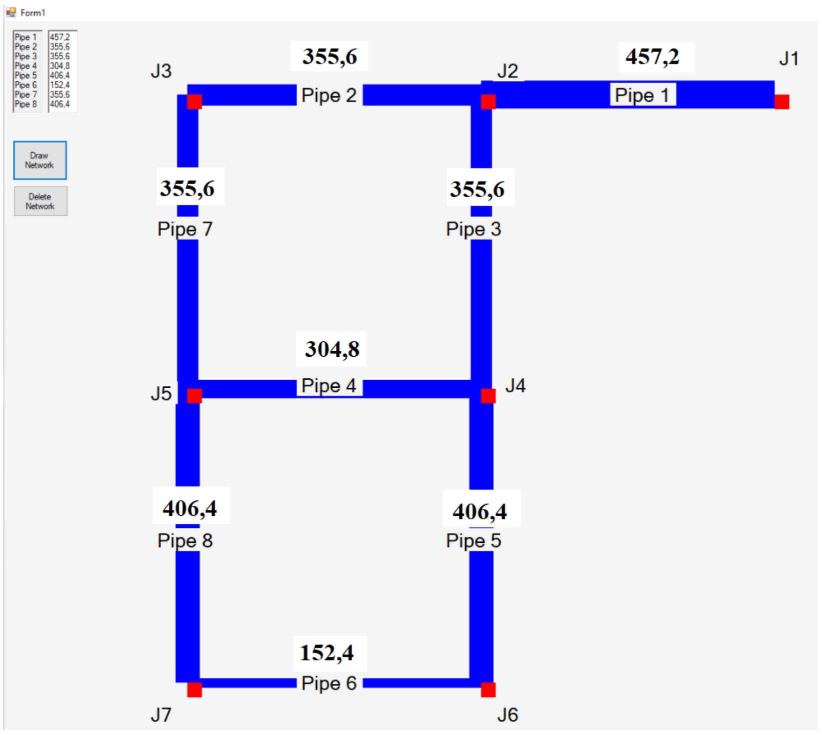
Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	304,8
Pipe 5	406,4
Pipe 6	254
Pipe 7	355,6
Pipe 8	406,4

Figure 5.15. Pipe diameters of two loop network end of the 8th iteration



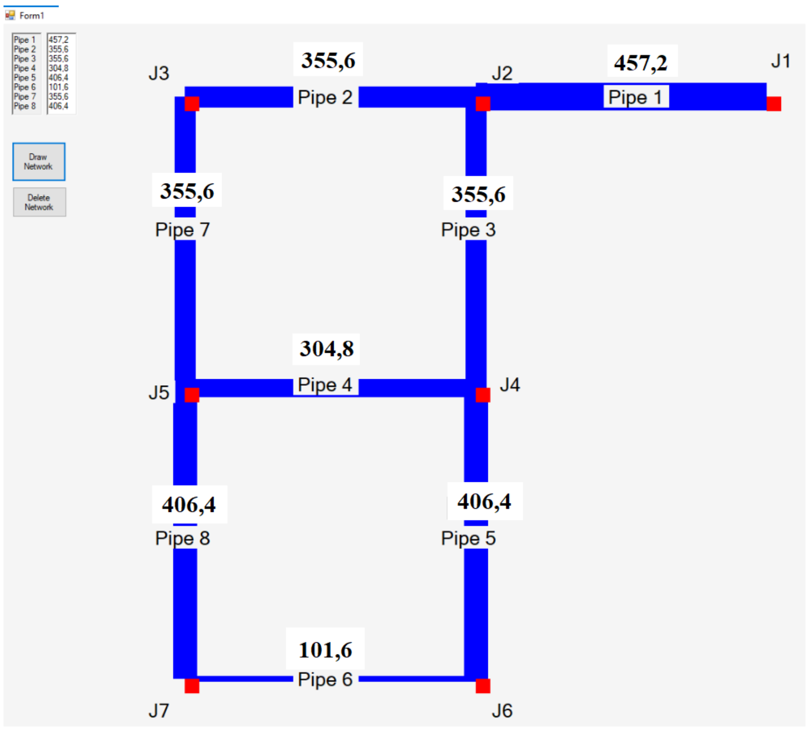
Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	304,8
Pipe 5	406,4
Pipe 6	203,2
Pipe 7	355,6
Pipe 8	406,4

Figure 5.16. Pipe diameters of two loop network end of the 9th iteration



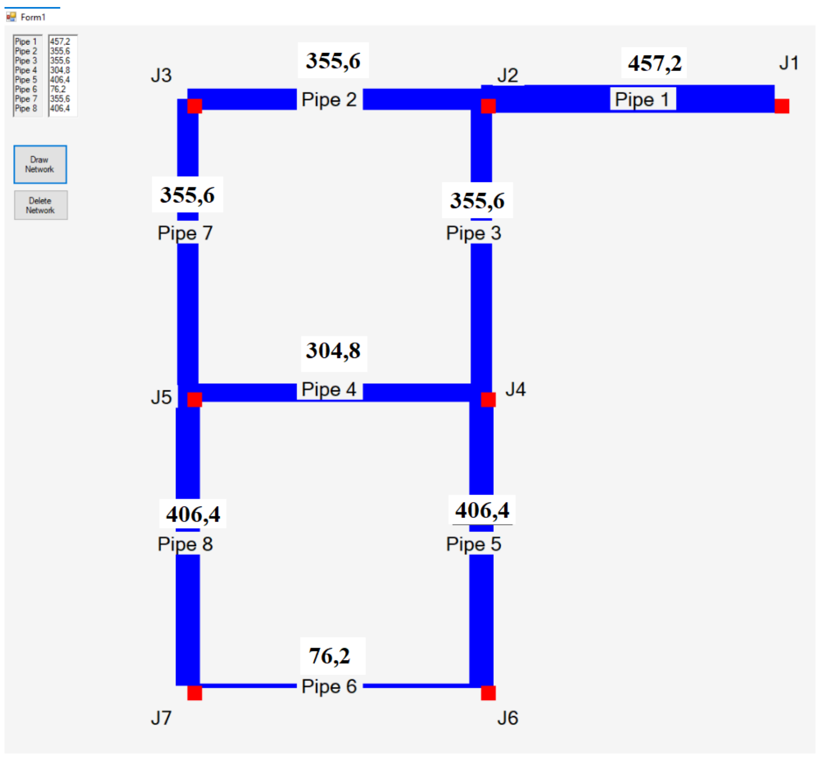
Pipe 1	457.2
Pipe 2	355.6
Pipe 3	355.6
Pipe 4	304.8
Pipe 5	406.4
Pipe 6	152.4
Pipe 7	355.6
Pipe 8	406.4

Figure 5.17. Pipe diameters of two loop network end of the 10th iteration



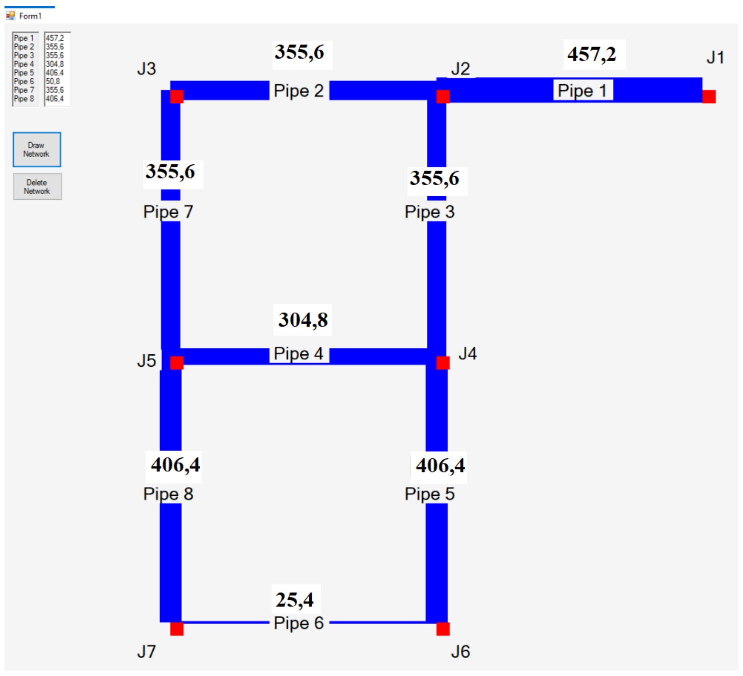
Pipe 1	457.2
Pipe 2	355.6
Pipe 3	355.6
Pipe 4	304.8
Pipe 5	406.4
Pipe 6	101.6
Pipe 7	355.6
Pipe 8	406.4

Figure 5.18. Pipe diameters of two loop network end of the 11th iteration


Form1

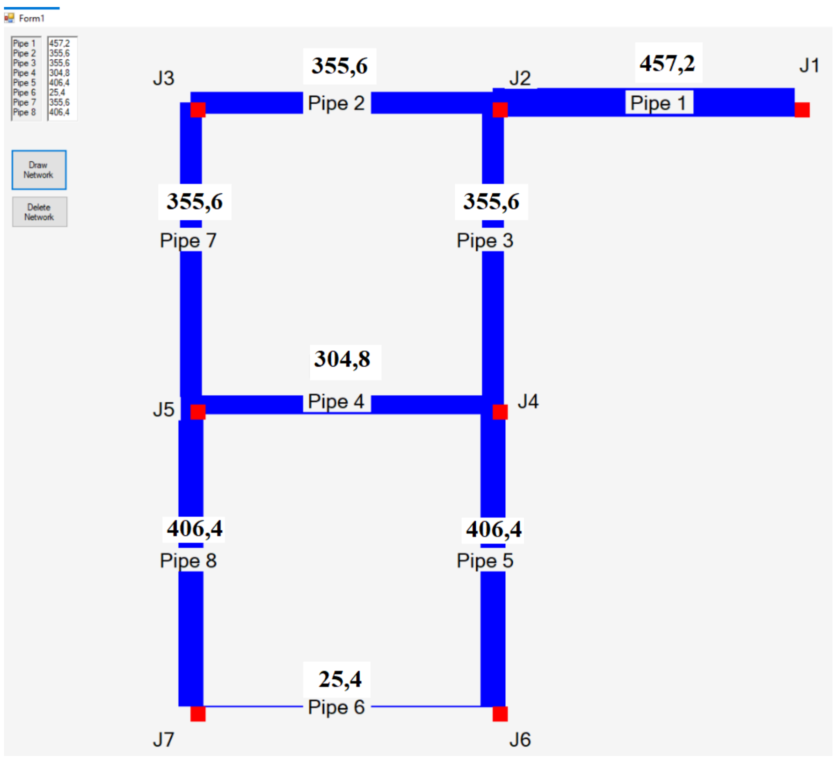
Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	304,8
Pipe 5	406,4
Pipe 6	76,2
Pipe 7	355,6
Pipe 8	406,4

Figure 5.19. Pipe diameters of two loop network end of the 12th iteration


Form1

Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	304,8
Pipe 5	406,4
Pipe 6	50,8
Pipe 7	355,6
Pipe 8	406,4

Figure 5.20. Pipe diameters of two loop network end of the 13th iteration



Pipe 1	457,2
Pipe 2	355,6
Pipe 3	355,6
Pipe 4	304,8
Pipe 5	406,4
Pipe 6	25,4
Pipe 7	355,6
Pipe 8	406,4

Figure 5.21. Pipe diameters of two loop network end of the 14th and final iteration

CHAPTER 6

CASE STUDY

In this chapter, on a existing network in Ankara (Yayla District) is studied using HALO which implements the heuristic approach presented by (Todini, 2000), whose usage and working logic are detailed in Chapter 5 and Chapter 6. At the end of this section, the optimal diameter for different resilience index value of the current Yayla District is indicated (Table 6.1).

6.1. Case Study: Ankara Network, N8.3 Yayla District

The N8.3 network is located in Ankara, Keçiören serves approximately 40,000 people (Gençoğlu, 2015). In order to increase the comprehensibility of this study, only a part of N8.3 pressure zone is Yayla District is considered in this chapter.

There are 158 demand nodes and 176 pipes in the model of Yayla network. The layout of Yayla DMA network is shown in Figure 6.1. The current pipe diameters of Yayla DMA is shown in Figure 6.2 (Gençoğlu, 2015).

The current Yayla DMA has a resilience index of 0.95 and a surplus head is 13.55 meters. Considering that the resilience index can take a value between 0 and 1 (Todini, 2000), it can be easily said that current Yayla DMA has very high reliability.

In this study, considering that Yayla DMA is overdesigned, the effect of increasing demand values on pipe diameters is also investigated. In other words, doubled and tripled nodal demands are assigned to Yayla DMA calculated by HALO. Consequently, pipe diameters and cost vs. resilience index values are shown at the end of this section (Figure 6.9, Figure 6.10 and Figure 6.11).

Another issue that must be addressed, current Yayla DMA pipe diameters can be found in the commercial Turkey market. However, in this section, Yayla DMA pipe diameters are designed by selecting from the diameter and price list given in (Todini, 2000).

The pipe diameters and prices inputted in HALO developed to implement the heuristic approach described by (Todini, 2000) can be changed, added or removed.

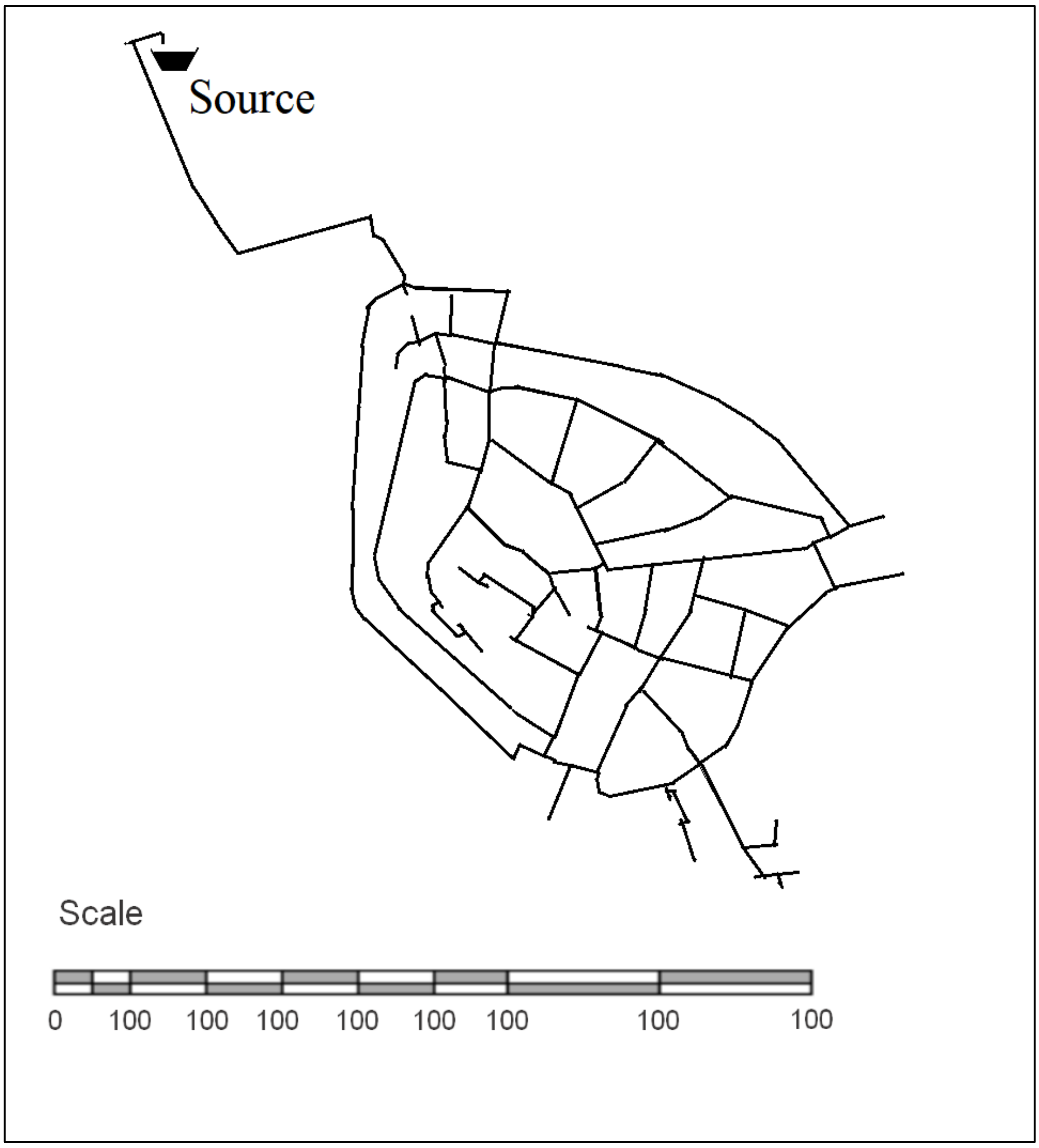


Figure 6.1. Layout of N8.3 Yayla Network

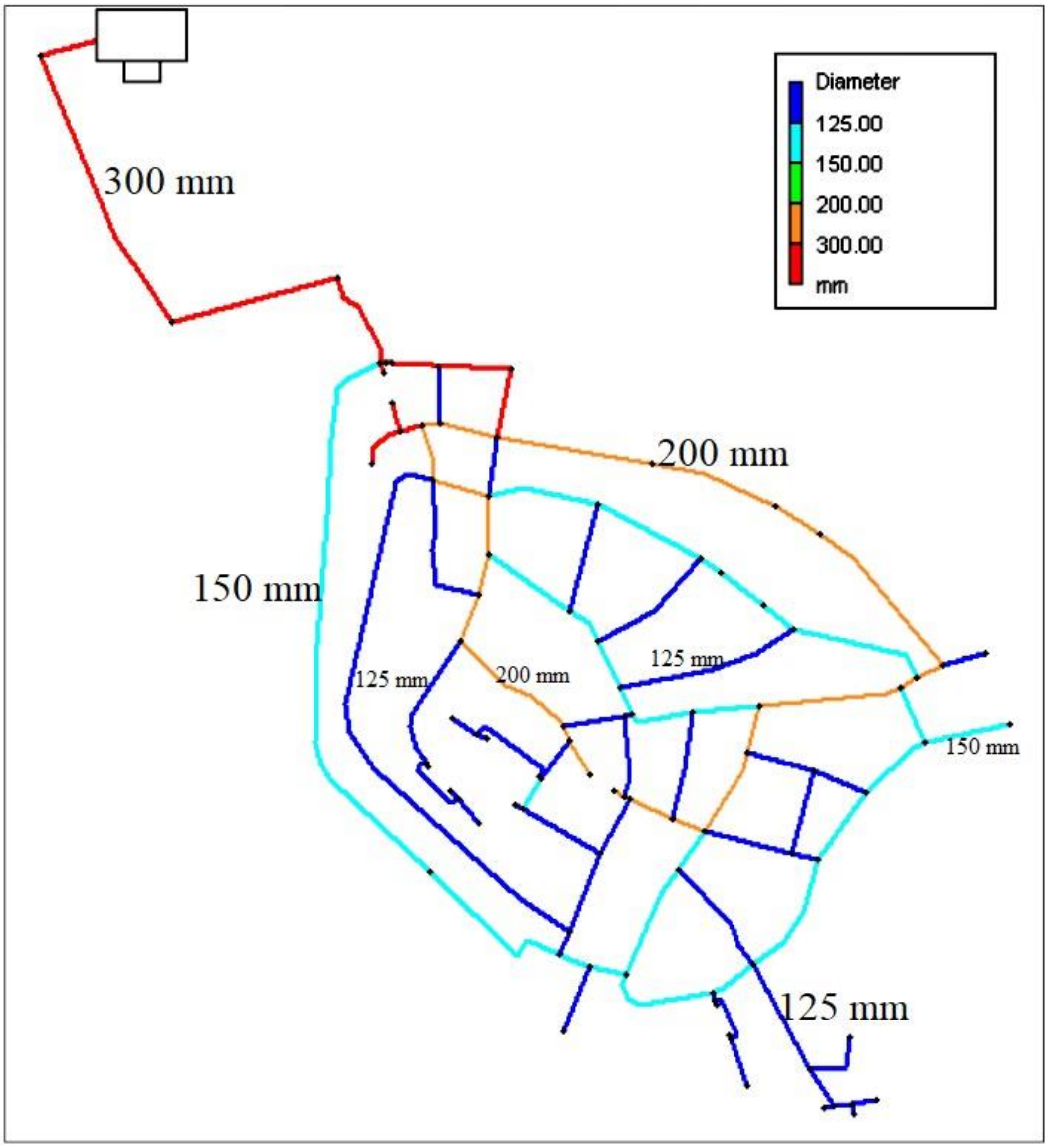


Figure 6.2. Pipe Diameters of Current Yayla DMA

6.2. Application of The Methodology on N8.3 Yayla District

N8.3 Yayla DMA is solved by HALO developed by coding the heuristic approach algorithm described by (Todini, 2000).

As described before (Figure 5.3 Step number 3), an initial resilience index must be defined. The user-defined resilience index specifies the minimum resilience index value of the WDN. It would be more practical to show the design engineer (or decision-maker) the graph of the costs corresponding to all resilience index values. Otherwise, the design engineer must apply the heuristic approach, one by one to the WDN for each resilience index value. Therefore, in HALO, cost values corresponding to all resilience index values are calculated between 0 and 1, with a precision of 1%. resilience vs cost is shown graphically (Figure 6.3).

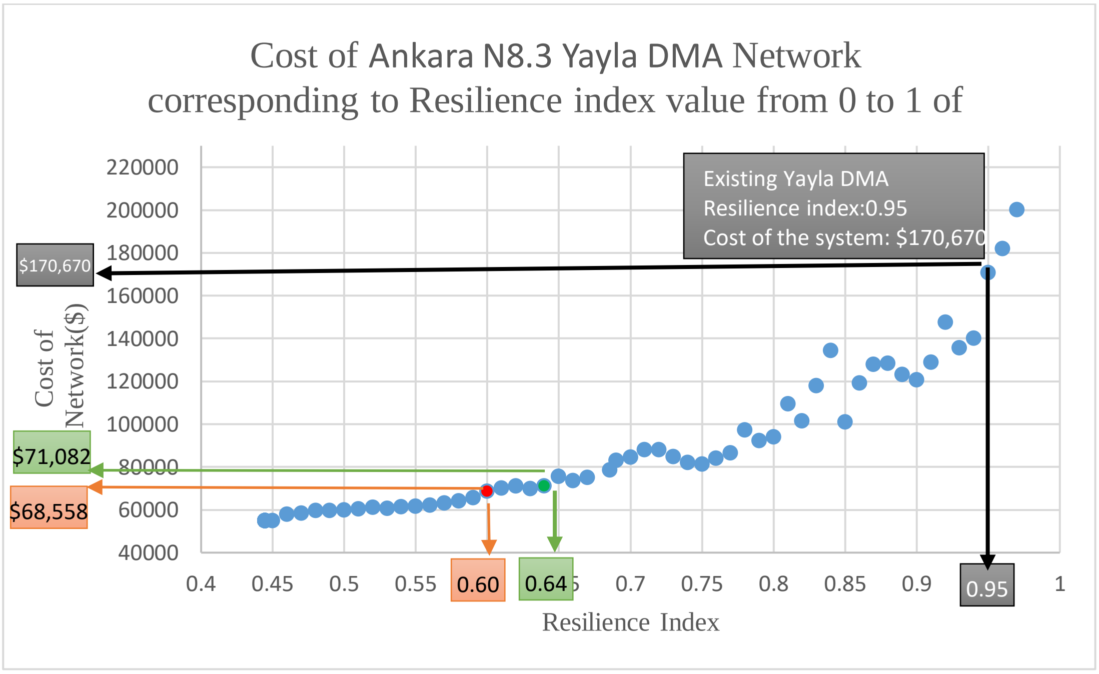


Figure 6.3. Costs corresponding to resilience index values between 0 and 1 of Ankara N8.3 Yayla DMA

Figure 6.3 shows the cost values corresponding to all resilience index values that Yayla DMA can take between 0 and 1. Figure 6.3 shows a solution cloud containing all possible reliability values for Yayla DMA. Figure 6.3 does not indicate which resilience index value is the optimal result to design the Yayla DMA. Figure 6.3 can show the rate of increase in budget to change network reliability to decision-makers. For example, two different resilience indices vs. cost values are shown with arrows in Figure 6.3 shows that decision-maker must pay \$ 68,558 to a system with a reliability index of 0.60. However, the purpose of Figure 6.3 (also this study) is to show that it can increase the reliability rate to 0.64 by increasing cost of this network only \$ 2,500. Decision-makers can see that a 3.5% increase in the cost versus a 6.7% increase in reliability can be achieved for this example.

Current Yayla DMA’s resilience index value is 0.95 (Figure 6.3) and the total cost of the network is \$ 170,670. Figure 6.3 shows that the minimum resilience index value of Yayla DMA is approximately 0.45. Minimum resilience index means that the diameters of Yayla are set to minimum pipe diameters that can meet nodal demands. If Yayla DMA’s pipe diameters are designed to provide, for example, 0.3 resilience index, then the required demands on the nodes cannot be satisfied.

The pipe diameters when Yayla DMA has resilience index values of 0.4, 0.5, 0.6 and 0.8 are shown in Figure 6.4, Figure 6.5, Figure 6.6 and Figure 6.7, respectively. It can be observed that the pipe diameters increase according to the resilience index values of Yayla DMA in the figures.

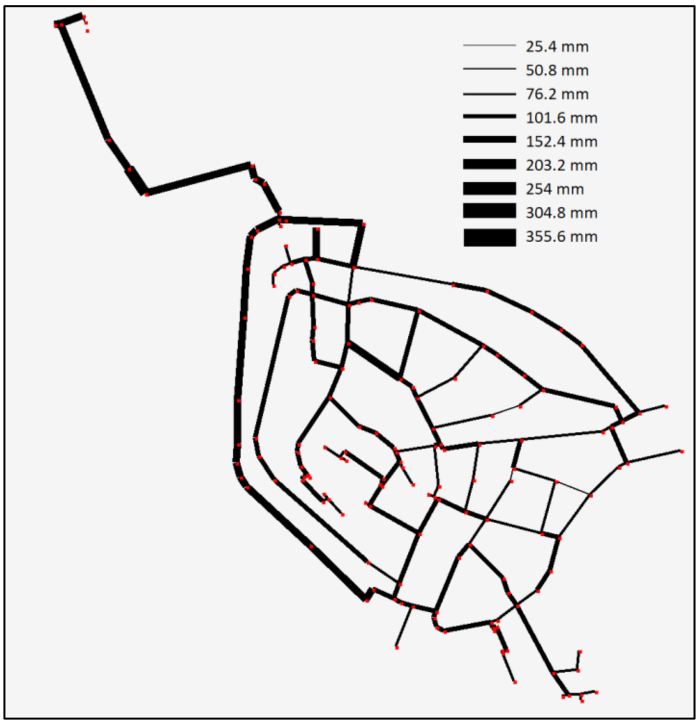


Figure 6.4. Diameters of the N8.3 Yayla DMA (Resilience index = 0.4)

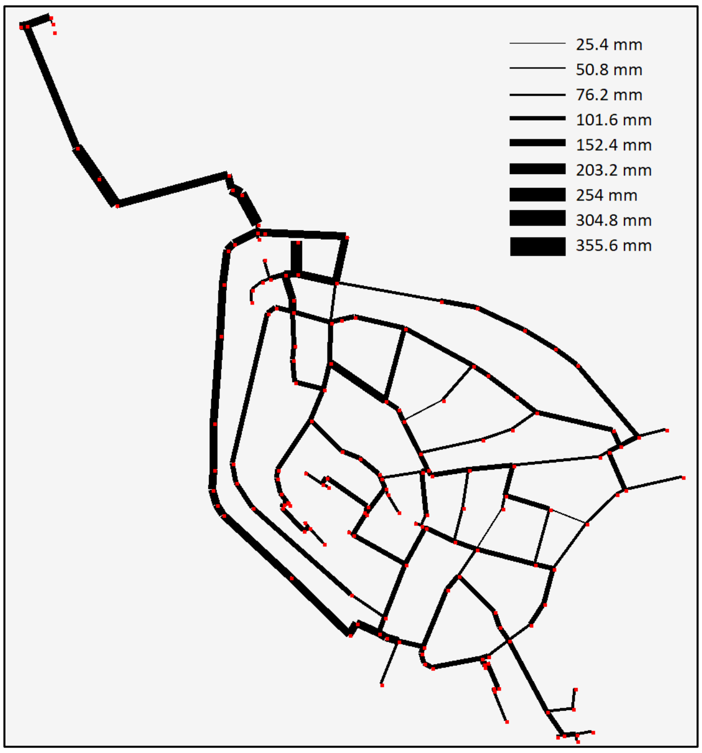


Figure 6.5. Diameters of the N8.3 Yayla DMA (Resilience index = 0.5)

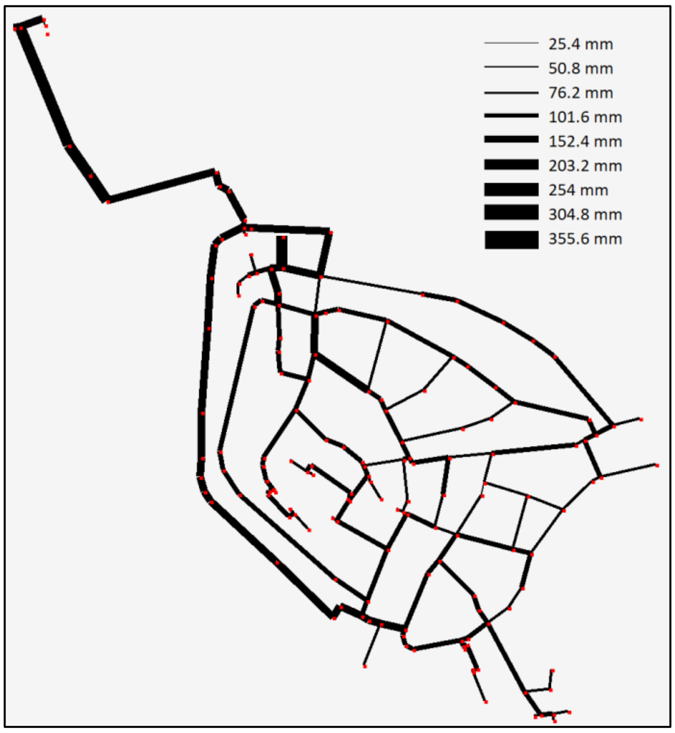


Figure 6.6. Diameters of the N8.3 Yayla DMA (Resilience index = 0.6)

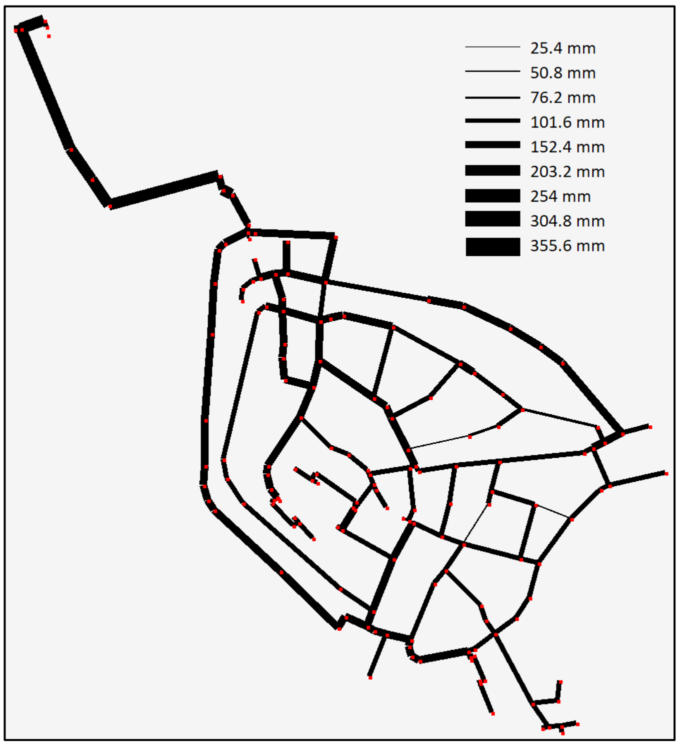


Figure 6.7. Diameters of the N8.3 Yayla DMA (Resilience index = 0.8)

Current Yayla DMA resilience index value is 0.95. It would not be wrong to say that current Yayla DMA is overdesigned. Because current demands in Yayla DMA are much lower than the capacity of the Yayla network. Although only pipe diameters are used as variables in this study, if the nodal demands of a WDN are increased, the effect on the pipe diameters of the Yayla network is investigated. The results obtained by using doubled and tripled demands of current Yayla demands are shown graphically in Figure 6.9, Figure 6.10 and Figure 6.11. Besides, Figure 6.8 shows the pipe diameters of Yayla network visually by using doubled demand in resilience index values of 0.4, 0.5, 0.6 and 0.8.

Considering Figure 6.9, Figure 6.10 and Figure 6.11, the increase in nodal demands in a WDN increases the minimum pipe diameters and costs. Besides, when Figure 6.9 and Figure 6.11 are considered simultaneous, the resilience index vs. cost value graph fluctuates less if the capacity of a WDN is approached in terms of demand.

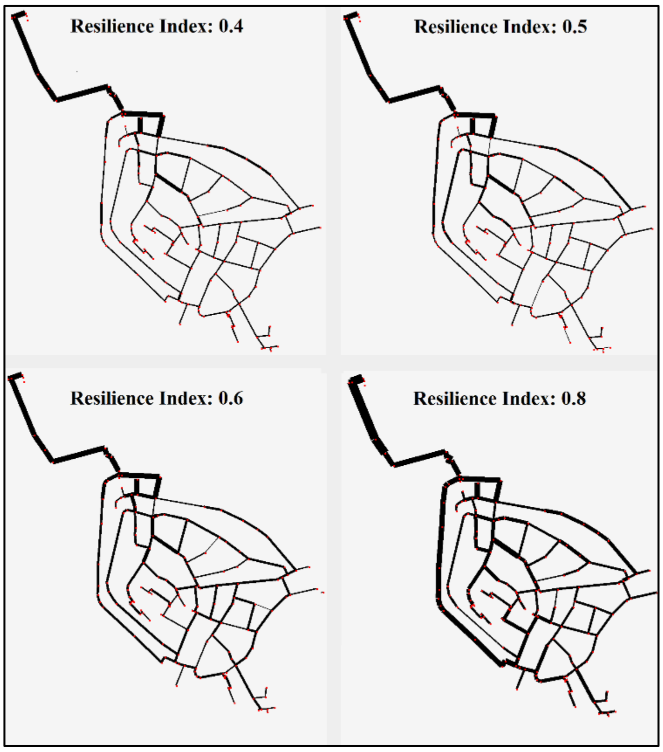


Figure 6.8. The system N8.3 Yayla DMA (demands are doubled)

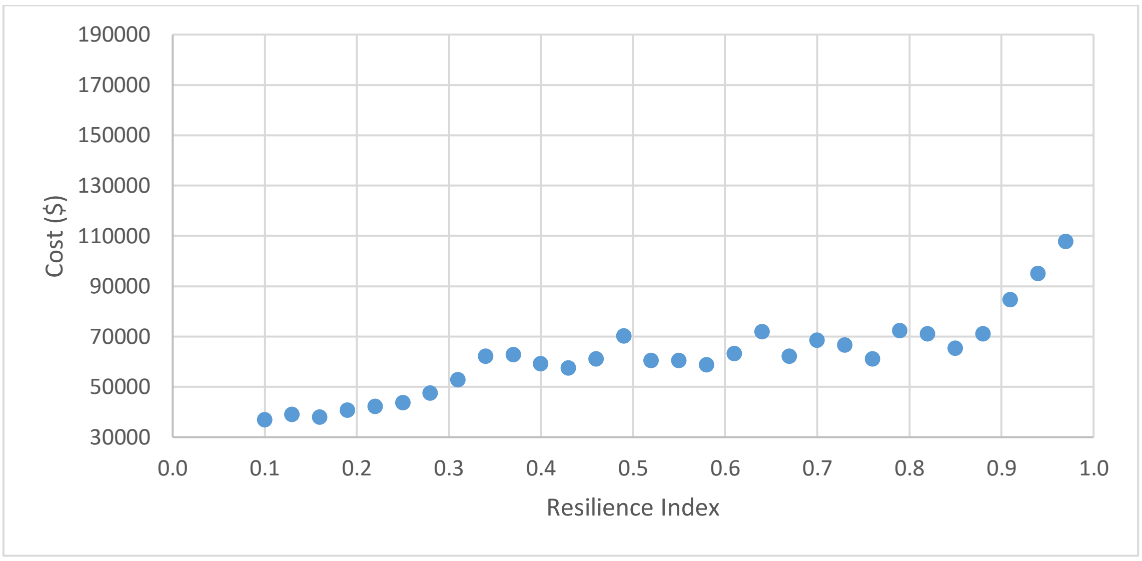


Figure 6.9. Pareto Set N8.3 Yayla

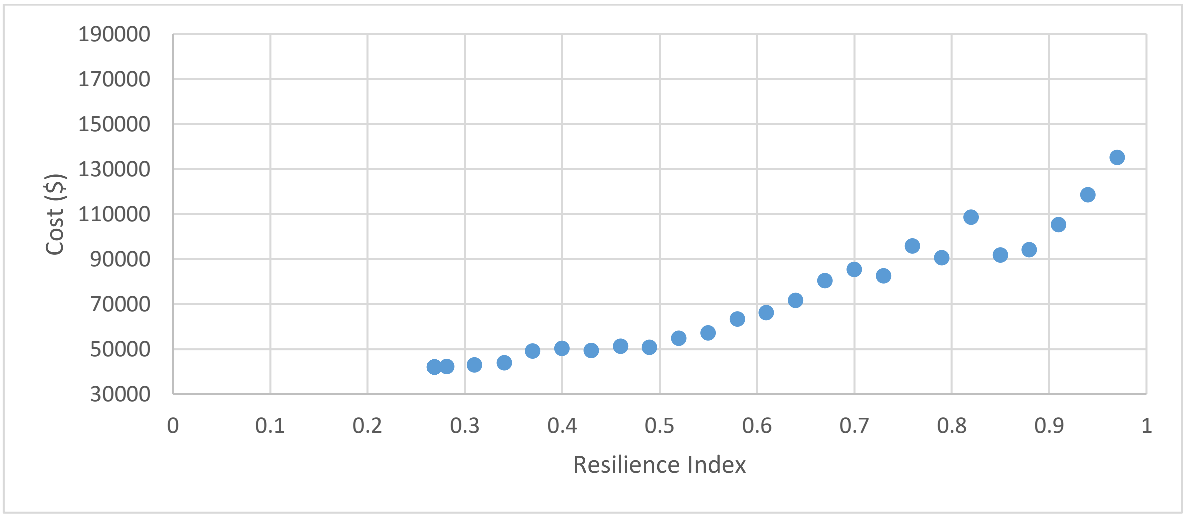


Figure 6.10. Pareto Set N8.3 Yayla (Demands are doubled)

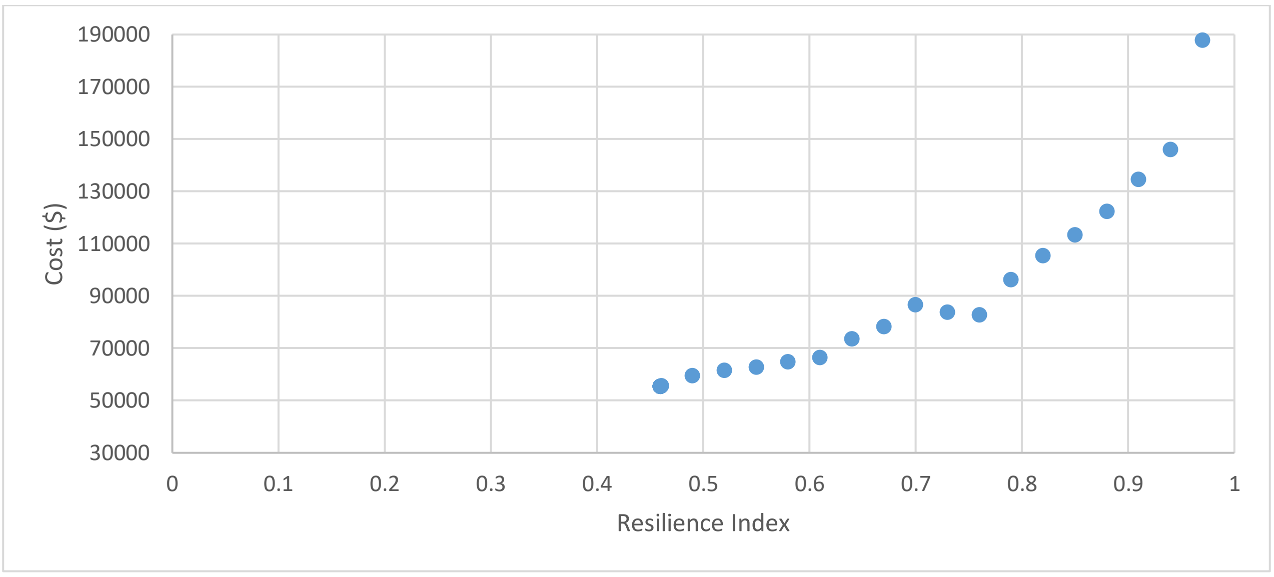


Figure 6.11. Pareto Set N8.3 Yayla (Demands are tripled)

6.3. Comparing the Results of Heuristic Approach with Existing Yayla Network

The first four columns of Table 6.1 represent the pipe diameters of the Yayla DMA with resilience index values of 0.4, 0.5, 0.6 and 0.8, respectively calculated by HALO.

It is noted that existing Yayla DMA pipes can be found in the Turkey market. However, the pipe list (Table 5.1) given by (Todini, 2000) is used in this study. It should also be considered that there are some limitations when designing pipe diameters of a WDN (e.g., minimum pipe diameter, diameters of transmission-line pipes). Since the methods given in (Todini, 2000) are used to develop HALO, there is no restriction in HALO on behalf of pipe diameters.

Table 6.1 shows that the current Yayla DMA pipe diameters are larger than even the diameters in the fourth column with a resilience index of 0.8, which is another proof that the current Yayla DMA is overdesigned.

Table 6.1. Comparison of current Yayla DMA pipe diameters vs HALO results.

Resilience Index: 0.4	Resilience Index: 0.5	Resilience Index: 0.6	Resilience Index: 0.8	Current Diameters
76.2	76.2	101.6	101.6	125
50.8	50.8	50.8	76.2	100
76.2	76.2	101.6	101.6	200
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	125
50.8	76.2	50.8	76.2	125
76.2	76.2	76.2	76.2	100
50.8	76.2	76.2	76.2	100
76.2	76.2	50.8	76.2	200
76.2	76.2	50.8	76.2	200
50.8	50.8	76.2	76.2	125
50.8	50.8	50.8	76.2	100
25.4	25.4	50.8	25.4	100
50.8	76.2	50.8	76.2	100
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	200
50.8	50.8	50.8	50.8	200
50.8	50.8	50.8	76.2	300
50.8	50.8	50.8	50.8	300
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100

Table 6.1.*(cont'd)* Comparison of current Yayla DMA pipe diameters versus HALO results.

Resilience Index: 0.4	Resilience Index: 0.5	Resilience Index: 0.6	Resilience Index: 0.8	Current Diameters
50.8	76.2	50.8	76.2	100
50.8	50.8	50.8	50.8	200
76.2	76.2	76.2	76.2	200
50.8	50.8	50.8	76.2	200
76.2	76.2	76.2	76.2	200
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	101.6	125
50.8	50.8	76.2	76.2	300
76.2	101.6	101.6	101.6	200
76.2	101.6	101.6	101.6	200
101.6	101.6	101.6	101.6	125
76.2	76.2	76.2	101.6	100
76.2	101.6	101.6	101.6	250
76.2	101.6	101.6	101.6	250
76.2	101.6	101.6	101.6	125
76.2	101.6	101.6	101.6	125
76.2	101.6	101.6	101.6	125
50.8	25.4	50.8	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	125
50.8	50.8	50.8	76.2	125
50.8	50.8	50.8	76.2	125
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	101.6	200
50.8	50.8	76.2	76.2	200
50.8	50.8	50.8	76.2	200
76.2	76.2	76.2	76.2	125

Table 6.1 *(cont'd)* Comparison of current Yayla DMA pipe diameters versus HALO results.

Resilience Index: 0.4	Resilience Index: 0.5	Resilience Index: 0.6	Resilience Index: 0.8	Current Diameters
76.2	76.2	76.2	76.2	125
25.4	50.8	25.4	76.2	200
50.8	25.4	50.8	25.4	200
76.2	76.2	76.2	101.6	125
50.8	50.8	76.2	76.2	125
50.8	76.2	76.2	76.2	125
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	50.8	125
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	101.6	200
50.8	50.8	50.8	76.2	100
50.8	50.8	76.2	76.2	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	50.8	76.2	100
50.8	50.8	50.8	76.2	200
76.2	76.2	76.2	76.2	200
76.2	76.2	76.2	76.2	200

Table 6.1.*(cont'd)* Comparison of current Yayla DMA pipe diameters versus HALO results.

Resilience Index: 0.4	Resilience Index: 0.5	Resilience Index: 0.6	Resilience Index: 0.8	Current Diameters
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	125
50.8	50.8	50.8	76.2	300
50.8	50.8	50.8	76.2	300
101.6	101.6	101.6	101.6	125
76.2	76.2	76.2	101.6	100
101.6	101.6	101.6	101.6	125
101.6	101.6	101.6	101.6	125
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
76.2	76.2	50.8	76.2	125
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	101.6	125
25.4	50.8	50.8	50.8	100
50.8	50.8	50.8	25.4	100
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
50.8	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
50.8	76.2	50.8	76.2	100
50.8	76.2	76.2	76.2	100
50.8	50.8	50.8	76.2	100
76.2	76.2	76.2	76.2	200
76.2	76.2	76.2	76.2	200
76.2	76.2	76.2	76.2	100
101.6	101.6	101.6	101.6	125
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	125

Table 6.1.*(cont'd)* Comparison of current Yayla DMA pipe diameters versus HALO results.

Resilience Index: 0.4	Resilience Index: 0.5	Resilience Index: 0.6	Resilience Index: 0.8	Current Diameters
76.2	76.2	76.2	76.2	100
50.8	76.2	76.2	76.2	100
76.2	76.2	76.2	76.2	100
101.6	101.6	101.6	101.6	125
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	76.2	100
101.6	101.6	101.6	101.6	125
76.2	76.2	76.2	76.2	100
50.8	50.8	76.2	76.2	100
101.6	101.6	101.6	101.6	125
101.6	101.6	101.6	101.6	125
101.6	101.6	101.6	101.6	125
101.6	101.6	101.6	101.6	125
101.6	101.6	101.6	101.6	125
50.8	50.8	50.8	76.2	125
50.8	50.8	50.8	76.2	100
101.6	101.6	101.6	152.4	300
101.6	101.6	101.6	101.6	300
101.6	101.6	101.6	101.6	300
101.6	101.6	101.6	101.6	300
101.6	101.6	101.6	101.6	300
101.6	101.6	152.4	152.4	300
101.6	101.6	101.6	152.4	500
25.4	25.4	25.4	76.2	500
101.6	101.6	101.6	101.6	500
101.6	152.4	101.6	152.4	300
101.6	152.4	152.4	152.4	300
152.4	152.4	152.4	152.4	300
101.6	152.4	101.6	101.6	300
50.8	50.8	76.2	76.2	300
76.2	76.2	76.2	76.2	100
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	100

Table 6.1.*(cont'd)* Comparison of current Yayla DMA pipe diameters versus HALO results.

Resilience Index: 0.4	Resilience Index: 0.5	Resilience Index: 0.6	Resilience Index: 0.8	Current Diameters
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	101.6	100
76.2	76.2	76.2	101.6	125
76.2	76.2	76.2	101.6	125
101.6	101.6	101.6	101.6	125
50.8	50.8	50.8	76.2	100
76.2	50.8	25.4	76.2	100
76.2	76.2	76.2	101.6	200
76.2	76.2	76.2	76.2	125
76.2	76.2	76.2	101.6	300
50.8	50.8	50.8	76.2	200
101.6	152.4	152.4	101.6	100

CHAPTER 7

DISCUSSION OF THE RESULTS

Resilience index is too sensitive to be used alone in WDN design. Because of the small changes in the resilience index value affect the pipe diameters in the network too much (Table 7.1). In Table 7.1, it is seen that the cost change is very high in even small change in the resilience index. Only a 1% increase in the resilience index, for example, from 0.44 to 0.45, leads to a 5% increase cost of the system. Aside from the cost increase, there is a big difference between Pipe-6 diameter for nearby resilience index values. In another case, the resilience index is 0.42 and 0.43, while the network costs are the same. However, in both networks, there is a huge difference between Pipe-6 pipe diameters. It should be considered that some of these results may be eliminated in Pareto analysis. Within the scope of this study, Pareto analysis is not performed to eliminate some results. Choosing the appropriate solution among the results is left to the decision-makers.

Table 7.1. *Pipe diameters of two-loop network resilience indices 0.41 to 0.44.*

Pipes Numbers	Results 1 (mm)	Results 2 (mm)	Results 3 (mm)	Results 4 (mm)	Results 5 (mm)
pipe 1	457.2	457.2	457.2	457.2	457.2
pipe 2	355.6	355.6	355.6	355.6	355.6
pipe 3	355.6	355.6	355.6	406.4	406.4
pipe 4	304.8	304.8	304.8	304.8	355.6
pipe 5	406.4	355.6	406.4	355.6	355.6
pipe 6	25.4	254	25.4	76.2	203.2
pipe 7	355.6	406.4	406.4	304.8	304.8
pipe 8	406.4	406.4	406.4	304.8	304.8
Resilience index	**0.41**	**0.42**	**0.43**	**0.44**	**0.45**
Surplus (m)	0.57	0.63	1.05	1.40	1.50
Cost of System ($)	542,000	572,000	572,000	498,000	523,000

The velocity check (Chapter 5.4. HALO Operation Steps - Step 9) of the heuristic approach described in (Todini, 2000) is never come into play in this case study. Flow velocities in Yayla DMA never approached 2 m/s.

The failure index (formula 4.3) described in (Todini, 2000) is never used during the heuristic approach method. Failure index refers to the degree of impact of pipes when the minimum required pressure cannot be achieved. Consequently, the failure index is not required if a network provides the minimum required pressure.

CHAPTER 8

CONCLUSIONS AND RECOMMENDATIONS

In this study, the heuristic approach described in (Todini, 2000) is applied to Ankara, Keçiören, Yayla DMA by using HALO. The resilience index of current Yayla DMA is 0.95. Considering that the resilience index value is between 0 and 1, the water distribution pipe diameters of the current Yayla DMA are overdesigned.

At the end of this study, Figure 6.3 graph is obtained. Figure 6.3 shows the cost of the Yayla DMA network versus the resilience index values increasing by 1% between 0 and 1. Each of the points in Figure 6.3 shows a valid and usable solution but does not indicate which solution is optimal. Figure 6.3 shows a solution cloud in terms of pipe diameters of Yayla DMA.

All results may not be available in the graph (Figure 6.3) with network diameters corresponding to resilience index values between 0 and 1. Due to pressure and speed constraints, some results may need to be eliminated. It is possible to eliminate some solutions by applying the ultimate Pareto curve technique to Figure 6.3 (Deb, 2001). The ultimate Pareto curve technique produces a Pareto optimal set. Which solution should be preferred can be determined by this method (Deb, 2001). However, within the scope of this thesis, all network solutions are shown to the decision-maker. In future studies, HALO can be developed to draw the ultimate Pareto curve to make it a choice.

The decision-maker to determine the dimensions of the Yayla DMA pipe diameters must make a trade-off between cost and reliability, according to Figure 6.3. In the example on the Figure 6.3 graph, a 3.5% increase in the design cost of a network with a resilience index of 0.60 can make the resilience index 0.64, which means a rise of

7% resilience index. Compared to the classical cost optimization technique, the 7% cost increase can increase the 86% resilience index (Table 5.7).

As the resilience index value presented in (Todini, 2000) increases, the surplus pressure value stored by the system against possible failures increases (Table 7.1 - Surplus values). In other words, the reliability of the system increases.

This study aims to provide the decision-maker with the opportunity to compare as described in the example above. At the end of the study, HALO is developed to enable the heuristic approach (Todini, 2000) to be implemented easily to real complex networks.

This multi-objective study aims to maximize reliability in case of failure in a WDN while minimizing the cost of the WDN. Reliability is defined as the resilience index (Todini, 2000). The resilience index is a good guideline for designing a WDN, if supported by some other indices to be well-controlled. The heuristic approach also works reasonably to design of a WDN. HALO that works successfully developed in this study can be used to apply Todini's heuristic approach to any desired WDN.

There may be some restrictions when designing pipe diameters for WDNs; for example, the minimum pipe diameters to be used on the streets and the main transmission line. The minimum pipe diameter of the Yayla DMA is 100 mm (Figure 6.2). However, the minimum pipe diameter used in this study is 25.4 mm (Table 5.1). In actual projects, pipe diameters are grouped to certain values in a WDN (see Figure 6.2). Since HALO is based on the algorithm and data described in (Todini, 2000), for this reason, no restrictions are imposed explained above. However, the restrictions mentioned can be easily implemented if desired in HALO.

It is not practical for adjacent pipes to have different diameter values. Therefore, when designing the WDN, the grouping of pipe diameters should be taken into consideration (Gençoğlu, 2007). In future studies, HALO can be developed for pipe grouping.

The resilience index represents the reliability of the system very well, but it needs to be supported by some other indices. There are some studies to improve the resilience index. For example (Creaco et al., 2016a), the relationship between loop diameter uniformity and pipes in the network is defined as a C coefficient of resilience index (Todini, 2000). In future studies, Creaco’s (2016a) loop uniformity method can be implemented to HALO.

In this study, the resilience index and failure index are introduced as a compact tool in terms of surplus head and delivered power of the water distribution network. In the formulations used in this study, the demand-driven modeling approach without leaks is considered. However, the resilience index and failure index formulations can be generalized to provide a more appropriate definition of leaking effect when dealing with pressure-driven running modeling (Creaco, 2016b).

In this study, a multi-objective optimization problem is considered with objective functions, resilience index and cost. However, components such as pumps, tanks and valves can also be found in a WDN. For example, many different parameters can be defined, such as operating times of pumps, location and number of valves and tanks. Although rising parameter numbers increases the computational load, allows for more realistic modeling of the network. In further studies, software that considers some other parameters on HALO can be developed.

REFERENCES

Alperovits, E. and Shamir, U. 1977. Design of optimal water distribution systems. Water Resources Research, 13(6), 885-900, doi:10.1029/WR013i006p00885.

Akyol, S. and Alataş, B. 2016. Güncel sürü zekâsi optimizasyon algoritmalari. Nevşehir Üniversitesi Fen Bilimleri Enstitü Dergisi, 5(2), 9-18.

Creaco, E., Fortunato, A., Franchini, M. and Mazzola, M. R. 2014. Comparison between entropy and resilience as indirect measures of reliability in the framework of water distribution network design. Procedia Engineering, 70, 379-388. doi:10.1016/j.proeng.2014.02.043.

Creaco, E., Franchini, M. and Todini, E. 2016a. The combined use of resilience and loop diameter uniformity as a good indirect measure of network reliability. Urban Water Journal, 13(2), 167–181. doi:10.1080/1573062X.2014.949799.

Creaco, E., Franchini, M. and Todini, E. 2016b. Generalized resilience and failure indices for use with pressure-driven modeling and leakage. J. Water Resour. Plann. Manage., 142(8). DOI: 10.1061/(ASCE)WR.1943-5452.0000656

Deb, K., Multi-Objective optimization using evolutionary algorithms. John Wiley & Sons, 2001, England.

Gençoğlu, G. 2007. Developing a methodology for the design of water distribution networks using genetic algoritm. METU Department of Civil Engineering, thesis of the degree of master of science, Ankara.

Gençoğlu, G. 2015. Optimizing pump schedule and valve characteristics of water distribution networks. METU Department of Civil Engineering, thesis of the degree of doctor of philosophy, Ankara. doi.org/10.1145/3132847.3132886.

Gessler, J., and Walski, T. M. 1985. Water distribution system optimization. Technical Report, Department of the US Army, Mississippi, USA.

Greco, R., Di Nardo, A., and Santonastaso, G. 2012. Resilience and entropy as indices of robustness of water distribution networks. Journal of Hydroinformatics, 14(3), 761–771. doi:10.2166/hydro.2012.037.

İller Bankası Anonim Şirketi, 2013. İçmesuyu tesisleri etüt, fizibilite ve projelerinin hazirlanmasina ait teknik şartname.

Kaya, S., and Kocaeli, L. 2016. Usage of pareto optimal in multi objective optimization problems. Social Sciences Research Journal, 5(2), 9–18.

Prasad, T. D., and Park, N. S. 2003. Multiobjective genetic algorithms for design of water distribution networks. Journal of Water Resources Planning and Management, 130(1), 73–82. doi:10.1061/(asce)0733-9496(2004)130:1(73).

Raad, D. N., Sinske, A. N. and Vuuren, J. H. 2010. Comparison of four reliability surrogate measures for water distribution systems design. Water Resources Research, 46(5), 1–11. doi:10.1029/2009wr007785.

Rossman, L. A., 1993. EPANET, Users manual. Risk reduction engineering laboratory, office of research and development, US Environmental Protection Agency, Cincinnati, Ohio.

Rossman, L. A., and Van Zyl, J. E. 2011. The open sourcing of EPANET. 19–28. doi:10.1061/41203(425)4.

Tanyimboh, T. T. and Templeman, A. B. 2000. A quantified assessment of the relationship between the reliability and entropy of water distribution systems. Engineering Optimization, 33(2), 179–199. doi:10.1080/03052150008940916.

Tanyimboh, T. T., Tietavainen, M. T. and Saleh, S. 2011. Performance assessment of water distribution systems. Water Science and Technology, 11(4), 437–443.

Todini, E. 2000. Looped water distribution networks design using a resilience index based heuristic approach. Urban Water, 2(2), 115–122. doi:10.1016/s1462-0758(00)00049-2.

Wang, Q., Savić, D. A., and Kapelan, Z. 2014. Hybrid metaheuristics for multi-objective design of water distribution systems. Journal of Hydroinformatics, Vol. 16, 165–177. doi:10.2166/hydro.2013.009

(Web 1) The Source code of HALO, 2019.

https://github.com/hibrahimates/HALO-software-Epanet-CSharp.

(Web 2) EPANET library for C#, 2014.

http://www.water-simulation.com/wsp/2014/02/25/epanet-class-for-c-sharp.

(Web 3) Programmer's Toolkits of EPANET, 2007.

http://epanet.de/developer/index.html.en.

APPENDICES

SOURCE CODE OF HALO

In this study, 12 class objects (for C#) is created for HALO

1. AnaMetod.cs: The class object in which the program is triggered.
2. Iterations.cs: The iteration procedure is defined in this class.
3. DonguClass.cs: It is the class in which iteration steps are managed.
4. NetworkOp.cs: The class in which the inp file is selected and the calculation outputs are managed.
5. NodeOp.cs: The class where all calculations related to nodes are made.
6. PipeOp.cs: The class in which all calculations for the pipes are made.
7. ReservoirOp.cs: The class in which all calculations for the reservoir are made.
8. ResIndexClass: The class in which the resilience index is calculated.
9. FailureIndexClass.cs: The class in which the failure index and Surplus index calculations are calculated.
10. Sabitler.cs: The class in which all constants used in the software are defined.
11. PublicListeler.cs: This is the class where the arrays used in the software are stored.
12. UnsafeNativeMethods.cs: The class which is contains EPANET calculation engine codes is downloaded from (Web 2).

A. AnaMetod.cs

```
using System;
using System.IO;
using System.Reflection;
using Epanet.Examples;
using System.Data.OleDb;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
using System.Windows.Forms;
using System.Drawing.Drawing2D;
namespace Epanet
{
    public partial class AnaMetod
    {
        private Button button1;
        private Label label1;
        private TextBox txt_resindex;
        private TextBox txt_minPressure;
        private Label label2;
        public static int sayac = 0;
        public static void Main(string[] args)
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
            Console.ReadLine();
        }
        public static float ToSingle(double value)
        {
            return (float)value;
        }
        private static void Error(string format, params object[]
args)
        {
            Console.ForegroundColor = ConsoleColor.Red;
            Console.Error.WriteLine(format, args);
            Console.Error.WriteLine("Aborting.");
            Console.ResetColor();
            Console.ReadLine();
        }
        private void InitializeComponent()
        {
            this.button1 = new System.Windows.Forms.Button();
            this.label1 = new System.Windows.Forms.Label();
            this.txt_resindex = new System.Windows.Forms.TextBox();
            this.txt_minPressure = new
System.Windows.Forms.TextBox();
            this.label2 = new System.Windows.Forms.Label();

            //
            // button1
            //
            this.button1.Location = new System.Drawing.Point(64,
108);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(156, 65);
            this.button1.TabIndex = 0;
            this.button1.Text = "Calculate";
            this.button1.UseVisualStyleBackColor = true;
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // label1
            //
            this.label1.AutoSize = true;
            this.label1.Location = new System.Drawing.Point(42, 29);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(156, 13);
            this.label1.TabIndex = 1;
            this.label1.Text = "User Defined Resilience Index :";
            //
            // txt_resindex
            //
            this.txt_resindex.Location = new
System.Drawing.Point(204, 26);
            this.txt_resindex.Name = "txt_resindex";
            this.txt_resindex.Size = new System.Drawing.Size(63,
20);
            this.txt_resindex.TabIndex = 2;
            this.txt_resindex.Text = "0,4122522";
            this.txt_resindex.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;
            //
            // txt_minPressure
            //
            this.txt_minPressure.Location = new
System.Drawing.Point(204, 61);
            this.txt_minPressure.Name = "txt_minPressure";
            this.txt_minPressure.Size = new System.Drawing.Size(63,
20);
            this.txt_minPressure.TabIndex = 4;
            this.txt_minPressure.Text = "29.5";
            this.txt_minPressure.TextAlign =
System.Windows.Forms.HorizontalAlignment.Center;
            //
            // label2
            //
            this.label2.AutoSize = true;
            this.label2.Location = new System.Drawing.Point(101,
61);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(97, 13);
            this.label2.TabIndex = 3;
            this.label2.Text = "Minimum pressure :";
            //
            // AnaMetod
            //
        }
        private void button1_Click(object sender, EventArgs e)
        {
            DonguClass.DonguGiris();
            DonguClass.DonguCalistir();
        }
    }
}
```

B. Iterations.cs

```
using System;
using System.Collections.Generic;
using System.Linq;

namespace Epanet
{
    class Iterations
    {
        public static bool Kontroller(int b) ///b. boru için kontrol
yapar\\\
        {
            float[] Velocityler = PipeOp.Get_Velocity();
            float f_index =
FailureIndexClass.Calculate_Failureindex();
            float r_index = ResIndexClass.Calculate_Resindex();
            PublicListeler.resres.Add(r_index);
            if (Velocityler.Max() <= 2 && r_index >
ResIndexClass.Kullanıcı_ResilienceIndex && f_index >= 0)
            {
                return true;
            }
            return false;
        }
        public static Dictionary<int, float>
Iteration_ILK(Dictionary<int, float> icListem)
        {
            PipeOp.Calculate_PowDis_ILK(icListem);
            var RankedList =
PublicListeler.Powdis_Dict_ILK.OrderByDescending(x =>
x.Value).ToList();
            foreach (var itemq in RankedList.ToList())
            {
                UnsafeNativeMethods.ENgetlinkvalue(itemq.Key + 1,
LinkValue.Diameter, out float valueDiam);
                PublicListeler.List_BoruvCap.Add(itemq.Key,
valueDiam);
            }
            RankedList.Clear(); // RANKEDLIST ICINI TEMIZLIYORUM
BURDA1
            NetworkOp.SaveAndRun();
            foreach (var FLitemler in PublicListeler.List_BoruvCap)
            {
                int a = (Array.FindIndex(Sabitler.Pipe_Diameters, x
=> x.Equals(FLitemler.Value)));
                float b = Sabitler.Pipe_Diameters[a - 1];
                UnsafeNativeMethods.ENsetlinkvalue(FLitemler.Key +
1, LinkValue.Diameter, b);  //Diameter Reduction


                if (Iterations.Kontroller(FLitemler.Key) == true)
                {
                    Console.WriteLine(FLitemler.Key + 1 + ". pipe
will be Reduced.....");
                    UnsafeNativeMethods.ENgetlinkvalue(FLitemler.Key
+ 1, LinkValue.Diameter, out float valueDiaa);

PublicListeler.AraList_kontrolSonrasi.Add(FLitemler.Key, valueDiaa);
                }
                else
                {
                    int d =
(Array.FindIndex(Sabitler.Pipe_Diameters, x =>
x.Equals(FLitemler.Value)));
                    float e = Sabitler.Pipe_Diameters[d];
                    UnsafeNativeMethods.ENsetlinkvalue(FLitemler.Key
+ 1, LinkValue.Diameter, e);
                    UnsafeNativeMethods.ENgetlinkvalue(FLitemler.Key
+ 1, LinkValue.Diameter, out float valueDiaa2);
                    Console.WriteLine(FLitemler.Key + 1 + ". pipe
will be remained SAME");

PublicListeler.AraList_kontrolSonrasi.Add(FLitemler.Key,
valueDiaa2);
                    NetworkOp.SaveAndRun();
                    PublicListeler.SabitListe.Add(FLitemler.Key,
valueDiaa2);       //boru çapı azalmayacağı için Sabit listeye
atanıyor.
                }
            }
            NetworkOp.OnlySave();
            Console.WriteLine("_____________________________");
            var yazdirilk = PublicListeler.List_BoruvCap.OrderBy(x
=> x.Key);
            foreach (KeyValuePair<int, float> kvp in yazdirilk)
            {
                UnsafeNativeMethods.ENsetlinkvalue(kvp.Key + 1,
LinkValue.Diameter, kvp.Value);
                Console.WriteLine("iterasyon sonucunda > Boru = {0}
= {1}", kvp.Key + 1, kvp.Value);
            }
            PublicListeler.List_BoruvCap.Clear();// List_BoruvCap
LISTESINI TEMIZLIYORUM BURDA
            return PublicListeler.AraList_kontrolSonrasi;
        }
        public static Dictionary<int, float>
Iteration(Dictionary<int, float> icListe)
        {
            PipeOp.Calculate_PowerDissipation(icListe);

            var RankedList =
PublicListeler.Powdis_Dict.OrderByDescending(x => x.Value).ToList();
            foreach (var itemq in RankedList.ToList())
            {
                UnsafeNativeMethods.ENgetlinkvalue(itemq.Key + 1,
LinkValue.Diameter, out float valueDiam);
```

```
                PublicListeler.List_BoruvCap.Add(itemq.Key,
valueDiam);
            }
            RankedList.Clear(); // RANKEDLIST ICINI TEMIZLIYORUM
BURDA
            PublicListeler.Powdis_Dict.Clear();//Powdis_Dict ICINI
TEMIZLIYORUM

            foreach (var FLitemler in PublicListeler.List_BoruvCap)
//kontroller ve azaltmalar
            {
                if
(PublicListeler.SabitListe.ContainsKey(FLitemler.Key))//Eğer test
edilen eleman Sabit listede varsa
                {
                    UnsafeNativeMethods.ENgetlinkvalue(FLitemler.Key
+ 1, LinkValue.Diameter, out float c2);

PublicListeler.AraList_kontrolSonrasi.Add(FLitemler.Key, c2);
                }
                else
                {
                    if (Array.FindIndex(Sabitler.Pipe_Diameters, x
=> x.Equals(FLitemler.Value)) == 0)
                    {
                        int a =
(Array.FindIndex(Sabitler.Pipe_Diameters, x =>
x.Equals(FLitemler.Value)));

UnsafeNativeMethods.ENsetlinkvalue(FLitemler.Key + 1,
LinkValue.Diameter, a);  //AZALTMAYI BURDA YAPIYORUM\\\
                    }
                    else
                    {
                        int a =
(Array.FindIndex(Sabitler.Pipe_Diameters, x =>
x.Equals(FLitemler.Value)));
                        float b = Sabitler.Pipe_Diameters[a -
1];//bu son boruya gelince saçmalıyor düzeltmemiz lazım

UnsafeNativeMethods.ENsetlinkvalue(FLitemler.Key + 1,
LinkValue.Diameter, b);  //AZALTMAYI BURDA YAPIYORUM\\\
                    }
                    if (Iterations.Kontroller(FLitemler.Key) ==
true)
                    {
                        Console.WriteLine(FLitemler.Key + 1 + ".
pipe will be Reduced.....");

UnsafeNativeMethods.ENgetlinkvalue(FLitemler.Key + 1,
LinkValue.Diameter, out float valueDiaa);

PublicListeler.AraList_kontrolSonrasi.Add(FLitemler.Key, valueDiaa);
                    }
                    else
                    {
                        int a =
(Array.FindIndex(Sabitler.Pipe_Diameters, x =>
x.Equals(FLitemler.Value)));
                        float c = Sabitler.Pipe_Diameters[a];

UnsafeNativeMethods.ENsetlinkvalue(FLitemler.Key + 1,
LinkValue.Diameter, c);
                        Console.WriteLine(FLitemler.Key + 1 + ".
pipe will be remained SAME");

PublicListeler.AraList_kontrolSonrasi.Add(FLitemler.Key, c);
                        NetworkOp.SaveAndRun();
                    }
                }
            }

            NetworkOp.SaveAndRun();
            foreach (var eleman in
PublicListeler.AraList_kontrolSonrasi)
            {
                if (eleman.Value ==
PublicListeler.List_BoruvCap[eleman.Key])   //&&
                {
                    if
(PublicListeler.SabitListe.ContainsKey(eleman.Key))
                    {
                        //null
                    }
                    else
                    {
                        PublicListeler.SabitListe.Add(eleman.Key,
eleman.Value);
                    }
                }
            }
            Console.WriteLine("_____________________________");
            var yazdir = PublicListeler.List_BoruvCap.OrderBy(x =>
x.Key);
            foreach (KeyValuePair<int, float> kvp in yazdir)
            {
                UnsafeNativeMethods.ENsetlinkvalue(kvp.Key + 1,
LinkValue.Diameter, kvp.Value);
                Console.WriteLine("iterasyon sonucunda > Boru = {0}
= {1}", kvp.Key + 1, kvp.Value);
            }
            PublicListeler.List_BoruvCap.Clear();// List_BoruvCap
LISTESINI TEMIZLIYORUM BURDA
            return PublicListeler.AraList_kontrolSonrasi;
        }

    }
}
```

C. DonguClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class DonguClass
    {
        public static void DonguGiris()
        {
            NetworkOp.StartSimulation();  // Networkü perform eder
            NetworkOp.SaveAndRun();
            NetworkOp.SaveAndRun();
            ResIndexClass.SetResindex();
            NetworkOp.SaveAndRun();
            PipeOp.Get_numofPipe();
            Console.WriteLine("Current Resilience index: " +
ResIndexClass.Calculate_Resindex());
            Console.WriteLine("Current Surplus value: " +
FailureIndexClass.Calculate_Failureindex());
        }
        public static void DonguCalistir()
        {
            int sayac = 0;

////////////////////////////////////////LOOP///////////////////////
//////////////////////////////////////////////////////////////////
            for (int i = 0; i < PipeOp.Get_numofPipe(); i++)
            {
                PublicListeler.ANALISTE.Add(i,
Sabitler.Pipe_Diameters[Sabitler.Pipe_Diameters.Length - 1]);//yani
558,8
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.InitSetting, 130);
            }
            Console.WriteLine(1 + ". Döngü Başlangıcı");
            PublicListeler.DonguListesi =
Iterations.Iteration_ILK(PublicListeler.ANALISTE).ToDictionary(entry
=> entry.Key, entry => entry.Value);
            Console.WriteLine("-------1. iterasyon sonucu Reilience
index: " + ResIndexClass.Calculate_Resindex());
            Console.WriteLine("-------1. iterasyon sonucu head
difference: " + FailureIndexClass.Calculate_Failureindex());
            Console.WriteLine("Hizlar ");
            var hizlar = PipeOp.Get_Velocity();
            for (int i = 0; i < PipeOp.Get_numofPipe(); i++)
            {
                Console.WriteLine(hizlar[i]);
            }
            PublicListeler.AraList_kontrolSonrasi.Clear();
            do
            {
                sayac = sayac + 1;
                Console.WriteLine("---------------------------------
---");
                Console.WriteLine(sayac + 1 + ". Döngü Başlangıcı");
                PublicListeler.DonguListesi =
Iterations.Iteration(PublicListeler.DonguListesi).ToDictionary(entry
=> entry.Key, entry => entry.Value);
                int say = sayac + 1;
                Console.WriteLine("-------" + say + ". iterasyon
sonucu Reilience index: " + ResIndexClass.Calculate_Resindex());
                Console.WriteLine("-------" + say + ". iterasyon
sonucu head difference: " +
FailureIndexClass.Calculate_Failureindex());
                Console.WriteLine("Hizlar ");
                var hizlar2 = PipeOp.Get_Velocity();
                for (int i = 0; i < PipeOp.Get_numofPipe(); i++)
                {
                    Console.WriteLine(hizlar2[i]);
                }
                PublicListeler.AraList_kontrolSonrasi.Clear();
            } while (PublicListeler.SabitListe.Count <
PublicListeler.ANALISTE.Count);

////////////////////////////////////////LOOP////////////////////////
////////////////////////////////////////////////////////////////////
/
            Console.WriteLine(" ");
            var sonhal = PublicListeler.SabitListe.OrderBy(x =>
x.Key).ToList();
            Console.WriteLine(" ");
            Console.WriteLine("-----------SON DURUMDA ÇAPLAR--------
-------");
            foreach (KeyValuePair<int, float> kvp in sonhal)
            {
                UnsafeNativeMethods.ENsetlinkvalue(kvp.Key + 1,
LinkValue.Diameter, kvp.Value);
                Console.WriteLine("Son hal > Boru = {0}, Çap = {1}",
kvp.Key + 1, kvp.Value);
            }
            Console.WriteLine(" ");
            float min = FailureIndexClass.Calculate_Failureindex();
            float rez = ResIndexClass.Calculate_Resindex();
            PipeOp.CostCalculation(PublicListeler.SabitListe);
            Console.WriteLine("Pressure-30m= " + min);
            Console.WriteLine("Resilience index: " + rez);
        }
    }
}
```

D. NetworkOp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class NetworkOp
    {
        public string f1 = "net10.inp";
        public static void StartSimulation()
        {
            string f1 = "net10.inp";          //Defining the INP file
            string f2 = "outDos.txt";
            string f3 = "repDos2.txt";
            UnsafeNativeMethods.ENopen(f1, f2, f3);     //Opening
the INP file
            UnsafeNativeMethods.ENgetcount(CountType.Link, out int
numofPipe);
            UnsafeNativeMethods.ENgetcount(CountType.Node, out int
numofNode);
            UnsafeNativeMethods.ENsolveH();  // Solving
hydraulically
        }
        public static void OnlySave()
        {
            UnsafeNativeMethods.ENsaveinpfile("net10.inp");
        }
        public static void SaveAndRun()
        {
            UnsafeNativeMethods.ENsaveinpfile("net10.inp");
            UnsafeNativeMethods.ENsolveH();  // Solving
hydraulically

        }
    }
}
```

E. NodeOp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class NodeOp
    {
        public static int Get_numofNode()
        {
            UnsafeNativeMethods.ENgetcount(CountType.Node, out int
numofNode);
            return numofNode;
        }
        public static float[] Get_NodeElev()
        {
            float[] NodeElevs = new float[Get_numofNode()];
            for (int i = 0; i < Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Elevation, out float NodeElev);
                NodeElevs[i] = NodeElev;
            }
            return NodeElevs;
        }
        public static float[] Get_NodeHead()
        {
            float[] NodeHeads = new float[Get_numofNode()];
            for (int i = 0; i < Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float NodeHead);
                NodeHeads[i] = NodeHead;
            }
            return NodeHeads;
        }

        public static float[] Get_Pressure()
        {
            float[] Pressures = new float[Get_numofNode()];
            for (int i = 0; i < Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Pressure, out float Pressure);
                Pressures[i] = Pressure;
            }
            return Pressures;
        }
        public static float[] Get_NodeDemand()
        {
            float[] NodeDemands = new float[Get_numofNode()];
            for (int i = 0; i < Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Demand, out float NodeDemand);
                NodeDemands[i] = NodeDemand * ((float)3.6);
            }
            return NodeDemands;
        }
        public static float[] Get_Hdesign()
        {
            float[] Hdesigns = new float[Get_numofNode()];
            for (int i = 0; i < Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Elevation, out float Hdesign);
                Hdesigns[i] = Hdesign + Sabitler.MinHead;
            }
            return Hdesigns;
        }
    }
}
```

F. PipeOp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class PipeOp
    {
        public static int Get_numofPipe()
        {
            UnsafeNativeMethods.ENgetcount(CountType.Link, out int
numofPipe);
            return numofPipe;
        }
        public static void Set_PipeDia(float Dia)
        {
            for (int i = 0; i < Get_numofPipe(); i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Dia);
            }
        }
        public static float[] Get_Flow()
        {
            float[] Flowlar = new float[Get_numofPipe()];
            for (int i = 0; i < Get_numofPipe(); i++)
            {
                UnsafeNativeMethods.ENgetlinkvalue(i + 1,
LinkValue.Flow, out float valueFlow);
                Flowlar[i] = Math.Abs(valueFlow);//Flow negatif
olamıyor ve birimi m3/h
            }
            return Flowlar;
        }
        public static float[] Get_HeadLoss()
        {
            float[] HeadLosslar = new float[Get_numofPipe()];
            for (int i = 0; i < Get_numofPipe(); i++)
            {
                UnsafeNativeMethods.ENgetlinkvalue(i + 1, 
LinkValue.HeadLoss, out float valueHloss);
                HeadLosslar[i] = valueHloss;
            }
            return HeadLosslar;
        }
        public static float[] Get_Velocity()
        {
            float[] Velocityler = new float[Get_numofPipe()];
            for (int i = 0; i < Get_numofPipe(); i++)
            {
                UnsafeNativeMethods.ENgetlinkvalue(i + 1, 
LinkValue.Velocity, out float valueVelocity);
                Velocityler[i] = valueVelocity;
            }
            return Velocityler;
        }
        public static void Calculate_PowDis_ILK(Dictionary<int, 
float> icListem)
        {
            int[] boruIndex1 = new int[icListem.Count];
            int[] boruIndex2 = new int[icListem.Count];
            float[] boruFiyat1 = new float[icListem.Count];
            float[] boruFiyat2 = new float[icListem.Count];
            float[] Cap1 = new float[icListem.Count]; //8 elemanlı 
dizi oluşturdu
            float[] Cap2 = new float[icListem.Count];
            float[] boruLenght1 = new float[icListem.Count];
            float[] boruLenght2 = new float[icListem.Count];
            float[] Pipe_Start_Point = new float[icListem.Count];
            float[] Pipe_End_Point = new float[icListem.Count];
            //listeyi sıralıyor..
            var iclistSirali = icListem.OrderBy(x => 
x.Key).ToList();
            Cap1 = iclistSirali.Select(z => z.Value).ToArray();
            //Boruların başlangıç ve bitiş noktalarını bulup dizi 
haline getiriyor..
            for (int i = 0; i < icListem.Count; i++)
            {
                UnsafeNativeMethods.ENgetlinknodes(i + 1, out int 
NodeS, out int NodeF);
                Pipe_End_Point[i] = NodeS;
                Pipe_Start_Point[i] = NodeF;
            }
            //boruların CAP1 ve CAP2 olarak uzunluk ve fiyatlarını 
buluyor..
            for (int k = 0; k < icListem.Count; k++)
            {
                UnsafeNativeMethods.ENgetlinkvalue(k + 1, 
LinkValue.Length, out float lenght1);
                boruLenght1[k] = lenght1;
            }
            for (int k = 0; k < icListem.Count; k++)
            {
                boruIndex1[k] =
Array.IndexOf(Sabitler.Pipe_Diameters, Cap1[k]);
                boruFiyat1[k] = Sabitler.Pipe_Cost[boruIndex1[k]] *
boruLenght1[k];
                Cap2[k] =
Sabitler.Pipe_Diameters[Array.IndexOf(Sabitler.Pipe_Diameters,
Cap1[k]) - 1];
                boruIndex2[k] =
Array.IndexOf(Sabitler.Pipe_Diameters, Cap2[k]);
                boruFiyat2[k] = Sabitler.Pipe_Cost[boruIndex2[k]] *
boruLenght1[k];
            }
            // Junctionların demandlerini alıyoruz.//BUNU SADECE 1
kere yapıyoruz...
            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Demand, out float DemandValue);
                if (DemandValue < 0)
                {
                    PublicListeler.Node_Demand_list.Add(i + 1, 0);
                }
                else
                {
                    float deger = DemandValue * (float)3.6;
                    PublicListeler.Node_Demand_list.Add(i + 1,
deger);
                }
            }
            //////////////////////////GAMA DURUMU İÇİN
(CAP2)//////////////////////
            // Start_Point noktası için HGL-Demand
            float[] Start_Point_HGL_CAP2 = new
float[icListem.Count];
            float[] End_Point_HGL_CAP2 = new float[icListem.Count];
            //Networke CAP2'yi atayıp sistemi analiz ediyoruz...
            for (int i = 0; i < icListem.Count; i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap2[i]);
            }
            NetworkOp.SaveAndRun();
            // HGL değerlerini çekiyoruz... node numarasına denk
gelen HGL leri listeliyoruz.
            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float HGL);
                PublicListeler.Node_HGL_list_CAP2.Add(i + 1, HGL);
            }
            //////////////////////////GAMA DURUMU İÇİN
(CAP1)//////////////////////
            // Start_Point noktası için HGL-Demand
            float[] Start_Point_HGL_CAP1 = new
float[icListem.Count];
            float[] End_Point_HGL_CAP1 = new float[icListem.Count];
            //Networke CAP1 i atayıp sistemi analiz ediyoruz...
            for (int i = 0; i < icListem.Count; i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap1[i]);
            }
            NetworkOp.SaveAndRun();
            // HGL değerlerini çekiyoruz...
            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float HGL);
                PublicListeler.Node_HGL_list_CAP1.Add(i + 1, HGL);
            }
            //power dissipationlar hesaplanacak ve Ratio bulunacak
            float[] PowDisp_1 = new float[icListem.Count];
            float[] PowDisp_2 = new float[icListem.Count];
            float[] Ratio = new float[icListem.Count];

            for (int t = 0; t < icListem.Count; t++)
            {
                UnsafeNativeMethods.ENgetlinknodes(t + 1, out int
NodeS, out int NodeF);
                if (PublicListeler.Node_HGL_list_CAP1[NodeS] >
PublicListeler.Node_HGL_list_CAP1[NodeF])
                {
                    PowDisp_1[t] =
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP1[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP1[NodeF]) / 3600;
                }
                else
                {
                    PowDisp_1[t] = -
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP1[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP1[NodeF]) / 3600;
                }

                if (PublicListeler.Node_HGL_list_CAP2[NodeS] >
PublicListeler.Node_HGL_list_CAP2[NodeF])
                {
                    PowDisp_2[t] =
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP2[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP2[NodeF]) / 3600;
                }
                else
                {
                    PowDisp_2[t] = -
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP2[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP2[NodeF]) / 3600;
                }

                Ratio[t] = -(boruFiyat2[t] - boruFiyat1[t]) /
((PowDisp_2[t] - PowDisp_1[t]) * Sabitler.g * 1000);
                PublicListeler.Powdis_Dict_ILK.Add(t, Ratio[t]);
            }
            for (int i = 0; i < icListem.Count(); i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap1[i]);
            }
            NetworkOp.SaveAndRun();
            PublicListeler.Node_HGL_list_CAP1.Clear();
            PublicListeler.Node_HGL_list_CAP2.Clear();
        }
        public static void
Calculate_PowerDissipation(Dictionary<int, float> icListe) //Burdaki
Dictionarylerde sorun yokz
        {
            PublicListeler.Powdis_Dict.Clear();//Her sefer yeniden
eleman ekleneceği için listeleri temizliyorum.
            int[] boruIndex1 = new int[icListe.Count];
            int[] boruIndex2 = new int[icListe.Count];
            float[] boruFiyat1 = new float[icListe.Count];
            float[] boruFiyat2 = new float[icListe.Count];
            float[] Cap1 = new float[icListe.Count]; //8 elemanlı
dizi oluşturdu
            float[] Cap2 = new float[icListe.Count];
            float[] boruLenght1 = new float[icListe.Count];
            float[] boruLenght2 = new float[icListe.Count];
            float[] Pipe_Start_Point = new float[icListe.Count];
            float[] Pipe_End_Point = new float[icListe.Count];
            var iclistSirali = icListe.OrderBy(x => x.Key).ToList();
            Cap1 = iclistSirali.Select(z => z.Value).ToArray();
            //Boruların başlangıç ve bitiş noktalarını bulup dizi
haline getiriyor..
            for (int i = 0; i < icListe.Count; i++)
            {
                UnsafeNativeMethods.ENgetlinknodes(i + 1, out int
NodeS, out int NodeF);
                Pipe_End_Point[i] = NodeS;
                Pipe_Start_Point[i] = NodeF;
            }
            for (int k = 0; k < icListe.Count; k++)
            {
                UnsafeNativeMethods.ENgetlinkvalue(k + 1,
LinkValue.Length, out float lenght1);
                boruLenght1[k] = lenght1;
            }
            for (int k = 0; k < icListe.Count; k++)
            {
                boruIndex1[k] =
Array.IndexOf(Sabitler.Pipe_Diameters, Cap1[k]);
                boruFiyat1[k] = Sabitler.Pipe_Cost[boruIndex1[k]] *
boruLenght1[k];

                if (PublicListeler.SabitListe.Count == 0)
                {
                    Cap2[k] =
Sabitler.Pipe_Diameters[Array.IndexOf(Sabitler.Pipe_Diameters,
Cap1[k]) - 1];
                }
                else
                {
                    if (PublicListeler.SabitListe.ContainsKey(k))
                    {
                        Cap2[k] =
Sabitler.Pipe_Diameters[Array.IndexOf(Sabitler.Pipe_Diameters,
Cap1[k])];
                    }
                    else
                    {
                        if (Array.IndexOf(Sabitler.Pipe_Diameters,
Cap1[k]) == 0)
                        {
                            Cap2[k] =
Sabitler.Pipe_Diameters[Array.IndexOf(Sabitler.Pipe_Diameters,
Cap1[k])];
                        }
                        else
                        {
                            Cap2[k] =
Sabitler.Pipe_Diameters[Array.IndexOf(Sabitler.Pipe_Diameters,
Cap1[k]) - 1];
                        }
                    }
                }
                boruIndex2[k] =
Array.IndexOf(Sabitler.Pipe_Diameters, Cap2[k]);
                boruFiyat2[k] = Sabitler.Pipe_Cost[boruIndex2[k]] *
boruLenght1[k];
            }
            for (int i = 0; i < icListe.Count(); i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap1[i]);//Eski haline getirdi sistemi
            }
            NetworkOp.SaveAndRun();
            ///////////////////////////GAMA DURUMU İÇİN
(CAP2)//////////////////////
            float[] Start_Point_HGL_CAP2 = new float[icListe.Count];
            float[] End_Point_HGL_CAP2 = new float[icListe.Count];
            //Networke CAP2'yi atayıp sistemi analiz ediyoruz...
            for (int i = 0; i < icListe.Count; i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap2[i]);
            }
            NetworkOp.SaveAndRun();
            // HGL değerlerini çekiyoruz...
            for (int i = 0; i < icListe.Count; i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float HGL);
                PublicListeler.Node_HGL_list_CAP2.Add(i + 1, HGL);
            }
            ///////////////////////////GAMA DURUMU İÇİN
(CAP1)//////////////////////
            // Start_Point noktası için HGL-Demand
            float[] Start_Point_HGL_CAP1 = new float[icListe.Count];
            float[] End_Point_HGL_CAP1 = new float[icListe.Count];

            //Networke CAP1 i atayıp sistemi analiz ediyoruz...
            for (int i = 0; i < icListe.Count; i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap1[i]);
            }
            NetworkOp.SaveAndRun();

            // HGL değerlerini çekiyoruz...
            for (int i = 0; i < icListe.Count; i++)
            {
                UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float HGL);
                PublicListeler.Node_HGL_list_CAP1.Add(i + 1, HGL);
            }

            ///Power Dissipation Calculation\\\
            float[] PowDisp_1 = new float[icListe.Count];
            float[] PowDisp_2 = new float[icListe.Count];
            float[] Ratio = new float[icListe.Count];


            for (int t = 0; t < icListe.Count; t++)
            {
                UnsafeNativeMethods.ENgetlinknodes(t + 1, out int
NodeS, out int NodeF);
                if (PublicListeler.Node_HGL_list_CAP1[NodeS] >
PublicListeler.Node_HGL_list_CAP1[NodeF])
                {
                    PowDisp_1[t] =
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP1[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP1[NodeF]) / 3600;
                }
                else
                {
                    PowDisp_1[t] = -
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP1[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP1[NodeF]) / 3600;
                }
                if (PublicListeler.Node_HGL_list_CAP2[NodeS] >
PublicListeler.Node_HGL_list_CAP2[NodeF])
                {
                    PowDisp_2[t] =
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP2[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP2[NodeF]) / 3600;
                }
                else
                {
                    PowDisp_2[t] = -
(PublicListeler.Node_Demand_list[NodeS] *
PublicListeler.Node_HGL_list_CAP2[NodeS] -
PublicListeler.Node_Demand_list[NodeF] *
PublicListeler.Node_HGL_list_CAP2[NodeF]) / 3600;
                }
                if (PowDisp_1[t] == PowDisp_2[t] || boruFiyat1[t] ==
boruFiyat2[t])//veya borufiyatlar eşitse
                {
                    PublicListeler.Powdis_Dict.Add(t, -9999999);
                }
                else
                {
                    Ratio[t] = -(boruFiyat2[t] - boruFiyat1[t]) /
((PowDisp_2[t] - PowDisp_1[t]) * Sabitler.g * 1000);
                    PublicListeler.Powdis_Dict.Add(t, Ratio[t]);
                }
            }
            for (int i = 0; i < icListe.Count(); i++)
            {
                UnsafeNativeMethods.ENsetlinkvalue(i + 1,
LinkValue.Diameter, Cap1[i]);
            }
            NetworkOp.SaveAndRun();
            PublicListeler.Node_HGL_list_CAP1.Clear();
            PublicListeler.Node_HGL_list_CAP2.Clear();
        }
        public static float CostCalculation(Dictionary<int, float>
icListem)
        {
            float toplamfiyat = 0;
            int[] boruIndex = new int[Get_numofPipe()];
            float[] boruFiyat = new float[Get_numofPipe()];
            float[] Cap = new float[icListem.Count];
            float[] boruLenght = new float[icListem.Count];

            var iclistSirali = icListem.OrderBy(x =>
x.Key).ToList();
            Cap = iclistSirali.Select(z => z.Value).ToArray();

            for (int k = 0; k < icListem.Count; k++)
            {
                UnsafeNativeMethods.ENgetlinkvalue(k + 1,
LinkValue.Length, out float lenght);
                boruLenght[k] = lenght;
            }
            for (int k = 0; k < Get_numofPipe(); k++)
            {
                boruIndex[k] =
Array.IndexOf(Sabitler.Pipe_Diameters, Cap[k]);
                boruFiyat[k] = Sabitler.Pipe_Cost[boruIndex[k]] *
boruLenght[k];
                toplamfiyat = toplamfiyat + boruFiyat[k];

                //  return toplamfiyat;
            }
            Console.WriteLine("Total Price :" + toplamfiyat);
            return toplamfiyat;
        }
    }
}
```

G. ReservoirOp.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class ReservoirOp
    {
        public static Dictionary<int, float> RezKotlari = new
Dictionary<int, float>();
        public static Dictionary<int, float> Rezbosaltim = new
Dictionary<int, float>();
        public float[] ResNodeElevations = new
float[RezKotlari.Count];
```

```
        public float[] ResNodeDischarges = new
float[RezKotlari.Count];

        public static String ChkNodeType = "Reservoir";
        public static float[] Get_ResElevations()
        {
            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodetype(i + 1, out
NodeType kontrol);
                if (kontrol.ToString() == ChkNodeType)
                {
                    UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float ResHead);
                    if (ResHead > 0)
                    {
                        if (RezKotlari.ContainsKey(i))
                        {
                            //null
                        }
                        else
                        {
                            RezKotlari.Add(i, ResHead);
                        }
                    }
                }
            }
            float[] ResNodeElevations = RezKotlari.Select(z =>
z.Value).ToArray();
            return ResNodeElevations;
        }
        public static float[] Get_ResNodeDischarges()
        {

            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodetype(i + 1, out
NodeType kontrol);

                if (kontrol.ToString() == ChkNodeType)
                {
                    UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Head, out float ResHead);
                    UnsafeNativeMethods.ENgetnodevalue(i + 1,
NodeValue.Demand, out float ResDischarge);
                    if (ResHead > 0)
                    {
                        if (Rezbosaltim.ContainsKey(i))
                        {
                            //null
                        }
                        else
                        {
                            Rezbosaltim.Add(i,
Math.Abs(ResDischarge));
                        }
                    }
                }
            }
            float[] ResNodeDischarges = Rezbosaltim.Select(z =>
z.Value).ToArray();
            return ResNodeDischarges;
        }
        public static float Calculate_TotalPowerofSystem()
        {
            float H_Q = 0;
            float[] resyukseklikleri =
ReservoirOp.Get_ResElevations();
            float[] rezbosaltimlari =
ReservoirOp.Get_ResNodeDischarges();
            for (int i = 0; i < resyukseklikleri.Length; i++)
            {
                float TotalPower = rezbosaltimlari[i] *
resyukseklikleri[i] * (float)3.6;
                H_Q = H_Q + TotalPower;
            }
            return H_Q;
        }
    }
}
```

H. ResIndexClass.cs

```
using System;
using System.IO;
using System.Reflection;
using Epanet.Examples;
using System.Data.OleDb;

using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Collections;
namespace Epanet
{
    class ResIndexClass
    {
        public static float Kullanıcı_ResilienceIndex;
        public static void SetResindex()
        {
            Console.WriteLine("Resilience Index: ");//Kullanıcıdan
Resilience index alınır
            if (Kullanıcı_ResilienceIndex < 1.0 &&
Kullanıcı_ResilienceIndex > 0.0)
            {
                //null
            }
            else
            {
                throw new Exception("Resilience Index must be
between 0-1 and use comma to define decimal");
            }
            Console.WriteLine("USER DEFINED RESILIENCE INDEX IS: " +
Kullanıcı_ResilienceIndex);
        }
        public static float Calculate_Resindex()
        {
            float T_demand_carp_h_eksi_hstar = 0;
            float T_demand_carp_Hdesign = 0;
            float[] Hdesignlar = NodeOp.Get_Hdesign();
            float[] Elevler = NodeOp.Get_NodeElev();
            float[] Demandler = NodeOp.Get_NodeDemand();
            float[] Pressurelar = NodeOp.Get_Pressure();
            float[] h_eksi_hstar = new
float[NodeOp.Get_numofNode()];
            float[] demand_carp_h_eksi_hstar = new
float[NodeOp.Get_numofNode()];
            float[] demand_carp_Hdesign = new
float[NodeOp.Get_numofNode()];
            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodetype(i + 1, out
NodeType kontrol);
                if (kontrol.ToString() !=
ReservoirOp.ChkNodeType)//reservuar değilse
                {
                    if (Demandler[i] > 0)
                    {
                        h_eksi_hstar[i] = Pressurelar[i] -
Sabitler.MinHead;
                        demand_carp_h_eksi_hstar[i] = Demandler[i] *
h_eksi_hstar[i];
                        T_demand_carp_h_eksi_hstar =
T_demand_carp_h_eksi_hstar + demand_carp_h_eksi_hstar[i];
                        demand_carp_Hdesign[i] = Demandler[i] *
Hdesignlar[i];
                        T_demand_carp_Hdesign =
T_demand_carp_Hdesign + demand_carp_Hdesign[i];
                    }
                }
            }
            double ResHesaplanmis = (T_demand_carp_h_eksi_hstar) /
(ReservoirOp.Calculate_TotalPowerofSystem() -
T_demand_carp_Hdesign);
            double rounded = Math.Round(ResHesaplanmis, 8);
            return (float)rounded;

        }
    }
}
```

İ. FailureIndexClass.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class FailureIndexClass
    {
        public static float Calculate_Failureindex()
        {
            float[] minDeger = new float[NodeOp.Get_numofNode()];
            float[] Pressurelar = NodeOp.Get_Pressure();
            for (int i = 0; i < NodeOp.Get_numofNode(); i++)
            {
                UnsafeNativeMethods.ENgetnodetype(i + 1, out
NodeType kontrol);
                if (kontrol.ToString() !=
ReservoirOp.ChkNodeType)//reservuar control
                {
                    minDeger[i] = Pressurelar[i] - Sabitler.MinHead;
                }
            }
            float minimumDeger = minDeger.Where(a => a > 0).Min();
            return minimumDeger;
        }
    }
}
```

J. Sabitler.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class Sabitler
    {
        public static float g = (float)9.81;
        public static float MinHead = (float)29.5;
        public static float[] Pipe_Diameters = new float[14]
        {
           (float)25.4  ,
           (float)50.8  ,
           (float)76.2  ,
           (float)101.6 ,
           (float)152.4 ,
           (float)203.2 ,
            (float)254.0 ,
            (float)304.8 ,
            (float)355.6 ,
            (float)406.4 ,
            (float)457.2 ,
            (float)508.0 ,
            (float)558.8,
            (float)609.6
        };

        public static float[] Pipe_Cost = new float[14]
        {
            2  ,
            5  ,
            8  ,
            11 ,
            16 ,
            23 ,
            32 ,
            50 ,
            60 ,
            90 ,
            130 ,
            170 ,
            300,
            550
        };
    }
}
```

K. PublicListeler.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Epanet
{
    class PublicListeler
    {
        public static List<float> resres = new List<float>();
        public static Dictionary<int, float> IterListesi = new
Dictionary<int, float>();
        public static Dictionary<int, float> ANALISTE = new
Dictionary<int, float>();
        public static Dictionary<int, float> Powdis_Dict = new
Dictionary<int, float>();
        public static Dictionary<int, float> Powdis_Dict_ILK = new
Dictionary<int, float>();
        public static Dictionary<int, float> AraList = new
Dictionary<int, float>();
```

```
        public static Dictionary<int, float> GuncelBoruListesi = new
Dictionary<int, float>();
        public static Dictionary<int, float> List_BoruvCap = new
Dictionary<int, float>();
        public static Dictionary<int, float> AraList_kontrolSonrasi
= new Dictionary<int, float>();
        public static Dictionary<int, float> dongu = new
Dictionary<int, float>();
        public static Dictionary<int, float> SabitListe = new
Dictionary<int, float>();
        public static Dictionary<int, float> DonguListesi = new
Dictionary<int, float>();
        public static Dictionary<int, float> Node_Demand_list = new
Dictionary<int, float>();
        public static Dictionary<int, float> Node_HGL_list_CAP1 =
new Dictionary<int, float>();
        public static Dictionary<int, float> Node_HGL_list_CAP2 =
new Dictionary<int, float>();

    }
}
```

L. UnsafeNativeMethods.cs

Unlicensed code of the UnsafeNativeMethods.cs class which is contains EPANET calculation engine codes is downloaded from (Web 2).