OPTIMUM DESIGN OF SLURRY PIPELINES

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

BY

BURHAN YILDIZ

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

DECEMBER 2009

Approval of the thesis

"OPTIMUM DESIGN OF SLURRY PIPELINES"

submitted by **Burhan Yıldız** in partial fullfillment of the requirements for the degree of **Master of Science in Civil Engineering** by,

Prof. Dr. Canan Özgen ______________
Dean, **Graduate School of Natural and Applied Sciences**

Prof. Dr. Güney Özcebe ______________
Head of Department, **Civil Engineering**

Assoc. Prof. Dr. Ayşe Burcu Altan Sakarya ______________
Supervisor, **Civil Engineering Department, METU**

Prof. Dr. Metin Ger ______________
Co-supervisor, **Civil Eng. Dept., T.C. İstanbul Kültür University**

Examining Committee Members:

Prof. Dr. Mustafa Göğüş ______________
Civil Engineering Department, METU

Assoc. Prof. Dr. Ayşe Burcu Altan Sakarya ______________
Civil Engineering Department, METU

Prof. Dr. Metin Ger ______________
Civil Engineering Department, T.C. İstanbul Kültür University

Assoc. Prof. Dr. İsmail Aydın ______________
Civil Engineering Department, METU

Dr. Yurdagül Kumcu ______________
General Directorate of State Hydraulic Works

Date: ______________

ABSTRACT

OPTIMUM DESIGN
OF
SLURRY PIPELINES

Yıldız, Burhan
M.S, Department of Civil Engineering
Supervisor: Assoc. Prof. Dr. Ayşe Burcu Altan Sakarya
Co-Supervisor: Prof. Dr. Metin Ger

December 2009, 58 pages

There exist various applications of transportation of slurries through pipelines all over the world. In the present study, the problem is formulated as a 'transportation problem' to determine the pipe diameters and amounts of slurry to be transported from the demand (production) points to the processing (factory) points. The minimization of the cost consisting of the pipe and energy cost terms is considered as the objective function to determine the stated decision variables. Pipe cost is given as the function of pipe diameters and the energy cost is defined as function of pipe diameters and slurry amounts. Energy cost is obtained by using the relation that is previously determined after the experimental studies made for the magnetic ore. The optimization method used in the study is Genetic Algorithms method. A commercially available software written in C language is used and modified for the present study. The proposed methodology to solve this nonlinear programming problem is applied to a transportation system and it is seen that the methodology made the complex, labor intensive equation solution process very convenient for the user.

Keywords: Slurry pipelines, Optimization, Genetic Algorithms

ÖZ

KATI MADDE TAŞINAN BORU HATLARININ OPTİMUM TASARIMI

Yıldız, Burhan
Yüksek Lisans, İnşaat Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Ayşe Burcu Altan Sakarya
Ortak Tez Yöneticisi: Prof. Dr. Metin Ger

Aralık 2009, 58 sayfa

Borularda katı madde taşınımının dünya genelinde çeşitli uygulamaları vardır. Mevcut çalışmada, problem bir ulaştırma problemi olarak, boru çaplarını ve üretim (talep) noktalarından tüketim (fabrika) noktalarına taşınan katı madde miktarını bulma üzerine formüle edilmiştir. Belirtilen karar değişkenlerinin bulunabilinmesi için, enerji ve boru hattı giderlerinden oluşan toplan maliyetin en küçüklenmesi amaç fonksiyonu olarak belirlenmiştir. Boru hattı maliyeti, boru çapının bir fonksiyonu olarak ele alınırken, enerji giderleri boru çapı ve taşınan katı madde miktarına bağlı bir fonksiyon olarak hesaplanmıştır. Enerji giderleri, daha önce manyetit konsantreleri için yapılan deneysel çalışmalardan elde edilen denklemler kullanılarak hesaplanmıştır. Optimizasyon yöntemi olarak Genetik Algoritma metodu kullanılmıştır. C dilinde yazılan bir serbest yazılım alınarak mevcut çalışma için uyarlanmıştır. Lineer olmayan programlama problemlerinin çözümü için önerilen bu yöntem, bir ulaştırma sistemi üzerinde uygulanmış ve uygulanan bu metodun karışık ve ağır zahmet gerektiren denklem çözümlerini kullanıcı için kolaylaştırdığı gözlemlenmiştir.

Anahtar Kelimeler: Katı-sıvı karışımı akımlar, optimizasyon, genetik algoritma

To My Family

ACKNOWLEDGMENTS

First and foremost, I would like to express my deepest gratitude to my research advisor Assoc. Prof. Dr. Burcu Altan Sakarya for her patience, guidance, encouragement and advice for this study.

I would like to extend my deepest gratitude to the co-supervisor of the study Prof. Dr. Metin Ger, for his perfect advises and support both academically and personally.

Another person that supports me during the study period is Assist. Prof. Dr. Şahnaz Tiğrek, I want to thank her for her guidance and advices.

My brother, Burak Yıldız, has helped me a lot on computer code development part, I would like to express my gratitude to him.

I would like to send special thanks to my friend and colleague, Korkut Demirbaş for his suggestions and comments on Genetic Algorithms.

I would like to thank all members of METU Hydraulics Laboratory for their support in the study period.

Moreover, I would like to express my deepest gratitude to my family who were always present and supportive in my life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

CW	Concentration by weight
CV	Concentration by volume
s_s	Specific gravity of solid material
ρ_w	density of water
V_m	Velocity of mixture
V_{cr}	Critical mixture velocity
W_s	Amount of solid material carried
d	Particle diameter of solid particle
D	Diameter of pipe
Q_m	Discharge of mixture
i	(h_l/l) slope of HGL
L	Length of the pipe
H_p	Head of the pump
CE	Unit cost of energy
γ_m	Specific weight of mixture
η	Efficiency of the pump
$C1$	Cost of energy per year
$C2$	Pipe cost

CHAPTER 1

INTRODUCTION

1.1 Problem Definition

Slurry pipeline method was first used in 1950s in order to transport ore materials. The method is found to be feasible in the situation of transporting materials to long distances. This type of transportation is obtained by making a mixture flow of water and the material to be transported. The pipelines are laid between the source of the material and the place where the material is to be processed. The concentrate of the ore is mixed with water and then pumped over long distances.

The hydraulics of slurry pipelines are somehow similar to the hydraulics of water pipelines. However, as it is a mixture flow, critical velocity of the particles in suspension should be computed. Also a head loss equation should be formulated to compute the required energy to the system. The head loss for mixture flows depends on pipe diameter, concentration of transported material and velocity of mixture. The related equations are obtained by the help of the information given in [1], which are presented in Chapter 3.

In this study the optimization of transportation of materials by means of slurry pipelines is handled. The problem is constructed as transporting the materials from multi production points to multi consumption points. The aim of the optimization problem in the study is to minimize the total cost. The constraint of the problem becomes the weight of the material to be transported. Pipe diameter and concentration values are selected as the decision variables of the problem. To consider a system with m production point and n consumption points, $m \times n$ pipelines should be laid between each point, then there exists $m \times n$ unknown pipe diameter and $m \times n$ unknown concentration value.

Due to the non-linearity in the equations to be used, the constructed optimization problem is a non-linear optimization problem. Also, there exists several unknowns as decision variables when considering multi production and multi consumption points. The method of optimization is selected by considering conditions. In the pre-studies, non-linear programming methods were tried. However, due to the high non-linearity of the problem, this method failed to converge to the optimal solution. Then it is decided to use the genetic algorithms method in solving the optimization problem. The main advantage of the genetic algorithms is that the algorithm always finds an answer whether the problem is highly nonlinear or not. Also, this solution can be improved applying various operators of the algorithm.

Previously, optimization of slurry pipeline is achieved by [1] where, the optimization process is done by trying each and every possible solutions by a computer code. Although there exists one production point and one consumption point in the study, the process is labor intensive. It can be estimated that, applying this method to a multi-inlet and multi-outlet system would not be practical. For example, if one wants to solve the real life study handled in the present study, which has 3 inlet and 3 outlet, over 20×10^{24} possible solutions should be examined within the boundaries of the decision variables in order to reach the optimal solution.

1.2 Literature Survey

In 1978, a feasibility study of constructing a slurry pipeline between Hasançelebi and İskenderun [1] is prepared by METU Hydraulics Laboratory upon the request of Ministry of Transportation. The optimization of the problem is solved within the report. The decision variable of the optimization problem was taken as pipe diameter and the problem was solved for the possible pipe diameters. The path of the pipeline was also a parameter in this study and the problem was solved several times for various weight of transported material. Besides, as the number of decision variable is one and the search space is relatively small, this process is achieved by a computer code written by the researchers. The computer code used in the study was based on a flow chart of the problem. Although the genetic algorithms method was not developed so far to apply on a project at the time, the search algorithm used in the study is somehow similar to the genetic algorithms method. In the search algorithm used, the possible solutions are inserted to the computer code, and the code evaluates each and every one systematically, then gives the best result as optimum. That algorithm did not need any operators, like the genetic algorithms have, to reach the result more quickly due to the small search space. Although, it can be estimated that, the algorithm used in the study may not work successfully for a problem having several decision variables and lots of possible results.

In 1979, two of the researchers of the previously explained study (Ger and Yücel) have written a congress paper on optimization of slurry pipelines with multi source points [5]. In this study an optimization problem is constructed by minimizing the cost of the system and constraining the weight of the transported material. The equations are converted to linear equations and solved by linear programming techniques.

In 1987 by Goldberg et al., pipeline optimization using genetic algorithms is done [6]. In this study, a pipeline problem with several pipes and several pumps is optimized. In this early study, the power and applicability of genetic algorithms method on pipeline problem is examined and the algorithm found very near-optimal results quickly. This study has been the first application of genetic algorithms to a water resources problem. After this study, many researchers followed the same way and applied genetic algorithms in water resources optimization problems.

In 1994 by Simpson et al., the comparison of optimization techniques for pipe optimization is studied [7]. In the study, a case of a small scaled pipe network is handled. The system is optimized to get a least-cost alternative by using methods; enumeration, non-linear optimization and genetic algorithms. By comparing the results, the researchers stated that; enumeration method gives good results on pipe networks with relatively few pipes, nonlinear optimization is an effective technique when applied to a small network expansions, however the method generates only one solution and discrete pipe sizes must be converted to a continuous function in order to use the method, and genetic algorithms method generates a whole class of alternative solutions close to the optimum. One of these solutions may be preferred to the optimum solution. Also, it is suggested that the genetic algorithm technique is ideally suited to discrete problems such as selection of commercially available pipe sizes.

Among the later studies, in 2000, Sharif et al., applied the genetic algorithms in optimizing multi reservoir systems [8]. In the study, a case is solved using dynamic programming methods and using genetic algorithms, then the obtained results are compared. It is concluded that the results obtained from both techniques are similar and genetic algorithms method has advantages over dynamic programming methods in practicability.

Another application of genetic algorithms on water resources is done by Farmani et al., in 2007 [9]. In the study, genetic algorithms is applied on optimum design of pressurized branched irrigation networks. The method is compared with linear programming methods, which is an alternative solution procedure for the problem. After applying genetic algorithms on the problem, it is concluded that, genetic algorithms are better in satisfying constraints and lowering the capital cost of the system.

1.3 Outline of the Thesis

In Chapter 2, Genetic Algorithms method is explained briefly. The operators of genetic algorithms and the parameters to be used in the study are discussed.

Then in Chapter 3, the problem is defined. At first, basic characteristics of slurry pipelines are given and hydraulic characteristics of this flow type is defined. Then, the optimization problem is constructed by means of objective function and constraints. At the end, the fitness function, to be used in Genetic Algorithms process, is defined.

In Chapter 4, the application of the problem is done for two situations. First, the method is applied to a single pipe system and then it is applied to a real life study. The results of these applications are presented in the same Chapter.

Finally, in Conclusion Chapter, a summary of the study is done, the gains from the study is explained and the way of improving the present study is stated.

CHAPTER 2

GENETIC ALGORITHMS

The selected optimization technique for the study is Genetic Algorithms, because the other methods were tried previously for the solution of the problem and an acceptable result could not be obtained due to nonlinearity of the equations. In this Chapter, Genetic Algorithms will be explained briefly.

Genetic Algorithms is a search algorithm based on the principles of natural genetics and evolution. It has been developed by John Holland in 1975. The first real world applications of Genetic Algorithms were designed in 80's after it has been studied theoretically until the early 80's.

The main advantages of solving an optimization problem by using genetic algorithms can be listed as; (i) it is reliable in either linear or non-linear problems, (ii) it gives always an answer to the problem, (iii) there exists many ways to improve the solution of genetic algorithms.

Genetic Algorithms is based on the main principle of the evolution, survival of the fittest. The good solution survive for the next generation, while the bad solution taken out from the population. The algorithm applies the same processes as the nature does in evolution. The schema of the Genetic Algorithms processes are shown in Figure 2.1. Mainly, the process starts with creating an initial population, which is the set of possible solutions. This process is done randomly. Then each individual is evaluated by use of fitness function, which is an evaluation function that determines which solutions are better than others. It is computed for each individual. Then a selection method is used to transfer the single individual to the next generation. The remaining individuals are selected for the mating pool. Crossover, mutation and elitism operators are applied to the individuals. The Genetic Algorithms operators will be explained in this Chapter, below. The parent individuals, entered into the operators constitutes the offspring. Then, finally by the insertion of the offspring into the

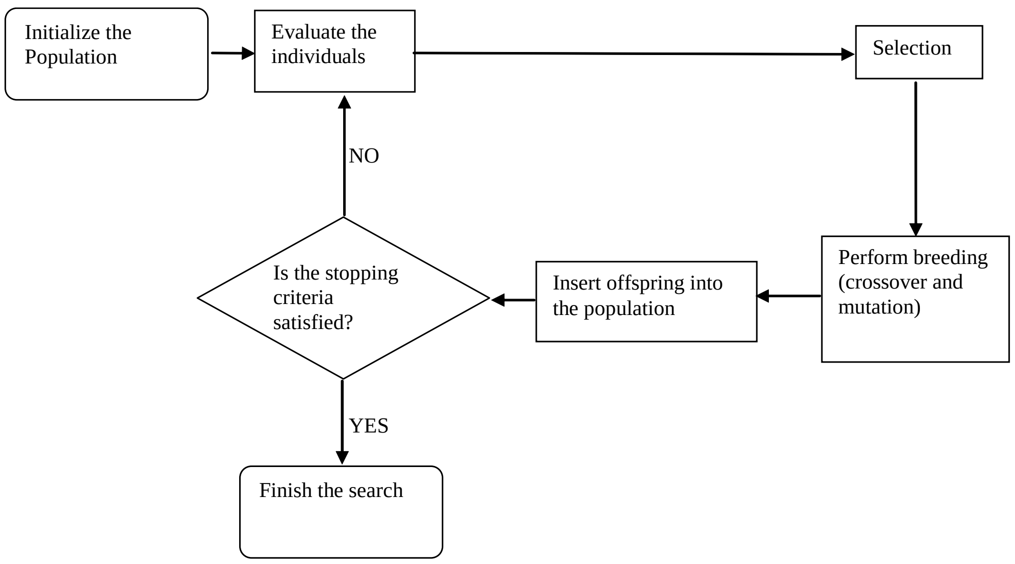
Initialize the Population
Evaluate the individuals
Selection
NO
Is the stopping criteria satisfied?
Insert offspring into the population
Perform breeding (crossover and mutation)
YES
Finish the search

Figure 2.1: Genetic Algorithms Processes

population, the new population for the next generation is obtained. If stopping criteria is satisfied after constructing the new population, the process ends and the individual in the population with the best fitness becomes the result. If the stopping criteria is not satisfied, the process continues until it is satisfied. The stopping criteria can be defined in two ways as; (i) stop when the generation number reaches the previously defined number, (ii) stop when the improvement of the average fitness over two generations is below a defined limit.

2.1 Operators of Genetic Algorithms

Mainly, there exists three Genetic Algorithms operators; selection operator selects the individuals to be transferred to the next generation, crossover and mutation operators changes the individuals to obtain better fitness values.

2.1.1 Selection

Selection operator transfers the single individual into the next generation based on fitness. Two possible selection methods are; (i) only the strongest survive because of their strong fitness (ii) some weak solutions also survive as some bad solutions might have good parts. Two main selection techniques are; roulette wheel selection and tournament selection.

In roulette wheel selection, each individual has a portion in the roulette wheel. The area of any individual in the roulette wheel is based on the fitness values. Then selection is done on roulette wheel, the probability of fittest individuals to be selected is higher than the weaker ones.

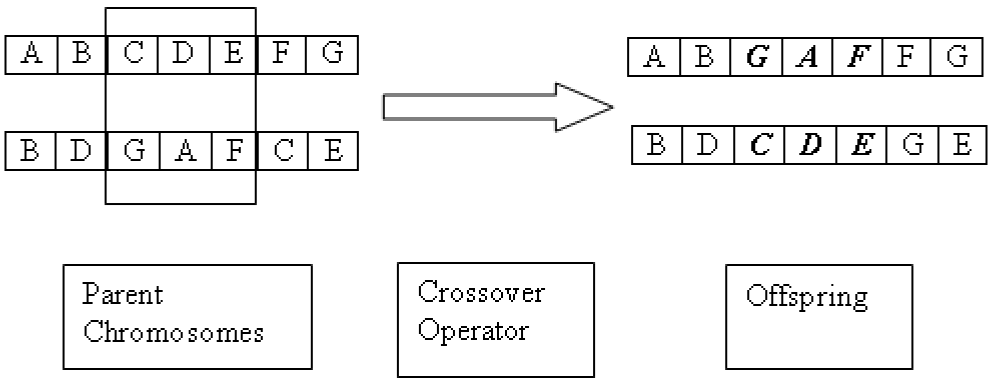

Figure 2.2: One-point Crossover Operator Schema

In tournament selection, subsets including n individuals are produced and the fittest of each subset are selected. n here is the tournament size. According to Prof. Kalyanmoy Deb from KANGAL (Kanpur Genetic Algorithms Laboratory), in the light of the explanations given in the Genetic Algorithms solver C Code he has developed, in minimization problems, tournament selection method should be used as selection operator.

2.1.2 Crossover

Crossover operator combines genetic material from two parents, in order to produce superior offspring. It is done to share the information among chromosomes. The aim is to combine the strong parts of the parents and get fitter offspring chromosomes. Mainly, one point and multi point crossover methods are applied in Genetic Algorithms. In Figure 2.2, basic one point crossover method is illustrated. The genes of the parent chromosomes change their places in order to create offspring chromosomes. In [10], it is stated that in most cases the crossover percentage is about 80%.

Generally two types of crossover methods are applied in real coded Genetic Algorithms; (i) simulated binary crossover (SBX-η), (ii) blend crossover (BLX-α). Both investigate the place of crossover site in different manner. In [11], it is stated that BLX-0.5 (α=0.5) performs better than the BLX operators with other α value. Besides, no recommendation for η value in SBX-η is found on articles.

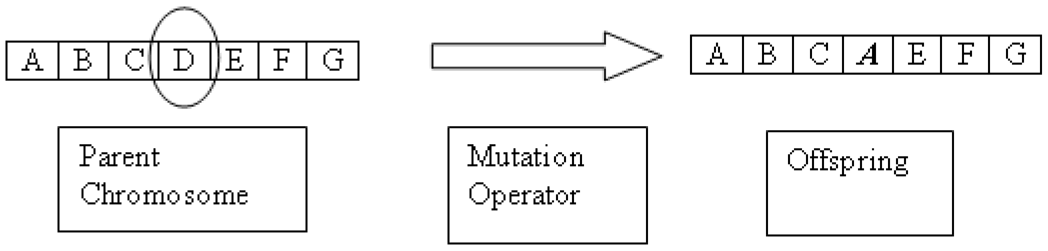

Figure 2.3: Mutation Operator Schema

2.1.3 Mutation

As the offspring is produced from one parent chromosome, while crossover is known as sexual reproduction in genetics, mutation is known as asexual reproduction. The idea of mutation is to reintroduce divergence into a converging population. It helps to extend the search space widespread. Mutation is performed on small part of population in order to avoid entering unstable state.

In Figure 2.3, an illustration shows how mutation operator works. The parent chromosome do not mate with any other chromosome to mutate. After applying the mutation operator to the parent chromosome, a gen or some genes of the chromosome change into another gen or genes. This operator is applied in order to reach a different path of the solution space.

2.2 Example Genetic Algorithms Solution by Hand

To explain genetic algorithms more clearly, a genetic algorithm solution by hand made in [4] is presented here.

The problem is to maximize $f(x) = x^2$, where x vary between 0 and 31. The procedure is presented in Table 2.1 and Table 2.2 The possible solutions are written in base 2 arithmetic in binary coding. (Second column in Table 2.1), the corresponding actual x values are given in third column, and corresponding $f(x)$ values are calculated. By dividing f_i to the $f_{ave.}$ expected counts are calculated and a comparison between individuals are done. Actual counts to be used in Roulette Wheel is found by using the expected count values. Elitism and mutation operators are not used in this problem. Only crossover operator is used. In Table 2.2, each individual is taken into mating pool for crossover operator. Then, randomly mates and the crossover sites are selected. In the table, mate numbers shows

Table 2.1: A Genetic Algorithm Solution by Hand[4]

String No.	Initial Population	x value	$f(x)$	Expected count	Actual count
1	01101	13	169	0.58	1
2	11000	24	576	1.97	2
3	01000	8	64	0.22	0
4	10011	19	361	1.23	1
Sum			1170	4.00	4.0
Average			293	1.00	1.0
Max			576	1.97	2.0

Table 2.2: A Genetic Algorithm Solution by Hand (continued)[4]

Mating Pool after Reproduction	Mate	Crossover Site	New Population	x value	$f(x)$
0110|1	2	4	01100	12	144
1100|0	1	4	11001	25	625
11|000	4	2	11011	27	729
10|011	3	2	10000	16	256
				Sum	1754
				Average	439
				Max	729

that the related string will match which string in crossover process and crossover site shows after which chromosome one-point crossover will be done. After the individuals reproduced, new population is obtained and the evaluation of the new population is done in the same manner as done in the first population. One generation of the algorithm is shown here. This processes continues until the stopping criteria is satisfied. As a result of this example, it is shown that the fitness is improved over two generations.

CHAPTER 3

PROBLEM FORMULATION

The aim of this study is to minimize the cost of the transportation of solid materials by pipe flow. First, the basic characteristics of solid-liquid mixture flows are investigated. Hydraulic computations are done to attain the hydraulic loss and critical velocity of mixture flow formulas. Then the energy required to overcome the head loss occurring during the pipe line and the cost of this amount of energy is calculated.

The cost of the system is reduced to a simpler state, as the total cost is taken as the sum of energy costs and pipe costs only. In [1], more detailed cost analysis can be found for a real world application.

In the present study, the pipe cost is calculated by the help of formula given in [12].

The constraints of the optimization problem, are the capacity and the demand of the production and the consumption points, respectively.

After defining the optimization problem, a genetic algorithms code based on C programming language written by Prof. Dr. Kalyanmoy Deb (1995) is adapted to the present study.

The detailed explanations of the above mentioned steps are presented in this chapter.

3.1 Hydraulic Definition

3.1.1 The Basics of Solid-Liquid Mixture Flows in Pipes

Solid-liquid mixture flows are mainly separated in two parts as settling and non-settling flows. In non-settling flow, as the particle diameters and particle settling velocity, i.e. fall velocity, are so small, the particles do not settle and flow in suspension with the effect of turbulence of the flow. Therefore, any cross-section of the pipeline carries the same volume of the particles. The flow behaves in a semi-homogeneous manner. On the contrary, in settling flow, the particle diameters and the settling velocities are higher. The turbulence effect of the flow is not enough to lift the particles from the ground. Therefore, the concentration of particles in suspension is not uniform over the cross section. From the application point of view, while non-settling flows can be treated as the flow of a new fluid, the settling flow characteristics depend on both solid and liquid characteristics individually. Furthermore, Durand and Condolios (1956) stated that if the volumetric concentration of solids exceeds 30 percent in settling flows, the flow characteristics start to behave like non-settling flows [1].

The border line between settling and non-settling flows is not very explicit [1]. Durand(1953) and Newitt(1955) stated that the range of particle settling velocity of $V = 0.60 \sim 1.50\ mm/s$ is the border line between two types of flows, such that the larger velocities are settling flows, and the lower velocities are non-settling flows. Additionally, in Wilson et al. [13], it is stated that, a slurry of small particles (less than 40 μm) in turbulent motion tends to behave in a homogeneous fashion. Also on the same source, the limit for pseudo-homogeneous flow (means that, there exists measurable decrease of concentration by height) are given for particles larger than 100 μm.

Also, Newitt(1955) and Aude(1971) obtained graphs showing the distinction between the flow types as a function of particle diameter, velocity of mixture and specific gravity of solid material, which is shown in Figure 3.1. As stated in Yücel et al. [1], ore materials, convenient for hydraulic transport, have particle diameters of $d_{50} = 40 - 100\ microns$ and specific gravity of $s_s = 4.6 - 6.2$ which corresponds the dashed regions in Figure 3.1. Therefore, it is concluded in Yücel et al. [1] that, the mixture flows convenient for hydraulic transport are semi-homogeneous and non-settling flows.

3.1.2 Head Losses in Non-Settling Solid-Liquid Mixture Flows

As mentioned above, in non-settling solid-liquid mixture flows, the particles are distributed uniformly in every cross-section of the pipe. Therefore, this mixture flow can be treated as a flow of a new fluid with different properties. The basic head loss formula for pipe flows is given in Equation 3.1.

$$H = i\,L \tag{3.1}$$

where, H is the head loss through the pipe, i is the head loss coefficient and L is the pipe length.

From the experiments [1], the following relation is obtained for ore concentrate mixture

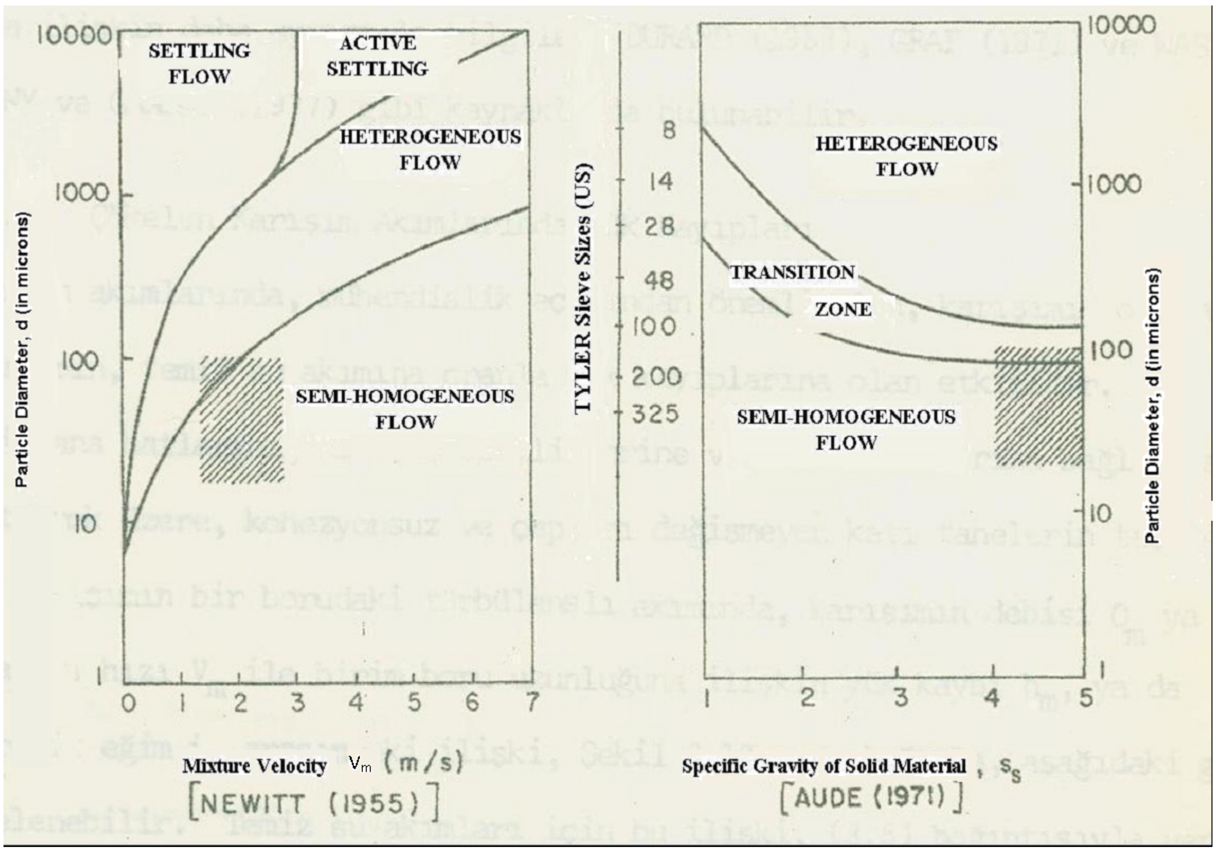

Figure 3.1: Classification of Mixture Flows [1]

flows convenient for slurry pipelining,

$$i = k\, C_V^x D^y\, V_m^z \tag{3.2}$$

where, C_V is the concentration by volume, D is the diameter and V_m is the velocity of mixture and k, x, y and z are the constants and determined after the laboratory studies.

First of all, to investigate the relation between i and V_m, Equation 3.2 can be written as,

$$i = k_1\, V_m^z \tag{3.3}$$

where, $k_1 = k\, C_V^x\, D^y$. The experiments were done for concentration by volume values of C_V= 0.20, 0.25 and 0.30 separately for the diameter values of D= 0.108 m, 0.159 m, 0.209 m, 0.263 m and 0.315 m [2]. Then researchers obtained the relation between i and V_m, which are shown in Figure 3.2, Figure 3.3 and Figure 3.4. As shown on these three graphs, z values are calculated for each diameter after the experimental studies and it is seen that all z values are close to each other. Then, the average of these values, z=1.77, will be used for all diameters in Equation 3.3 [1].

After determining the exponent of V_m, k_1 values are calculated as $k_1 = i/V_m^{1.77}$ from Equation 3.3. After k_1 values of all experiment sets are calculated and according to equation below,

$$k_1 = k_2\, D^y \tag{3.4}$$

where,$k_2 = k\, C_V^x$.

$k_1$versus D graphs are drawn for various C_V values as shown in Figure 3.5. It is seen from the equations of the lines that, the slopes of all lines, the exponent of diameter in Equation 3.2 i.e. y value, are the same. k_2 values are also obtained as 0.001074, 0.001284 and 0.001487 for C_V of 0.2, 0.25 and 0.3, respectively.

As $k_2 = k\, C_V^x$ and k_2 and corresponding C_V values are known, a graph showing the relation between k_2 and C_V was obtained (Figure 3.6). As shown on Figure 3.6, two appropriate equations were fitted on the graph. The equation of exponential form (i.e. $k_2 = 0.0039\, C_V^{0.803}$) is chosen, which makes the final version of Equation 3.2 to become,

$$i = 0.0039 C_V^{0.803} D^{-1.25} V_m^{1.77} \tag{3.5}$$

where, i is in m/m, D is in m and V_m is in m/s.

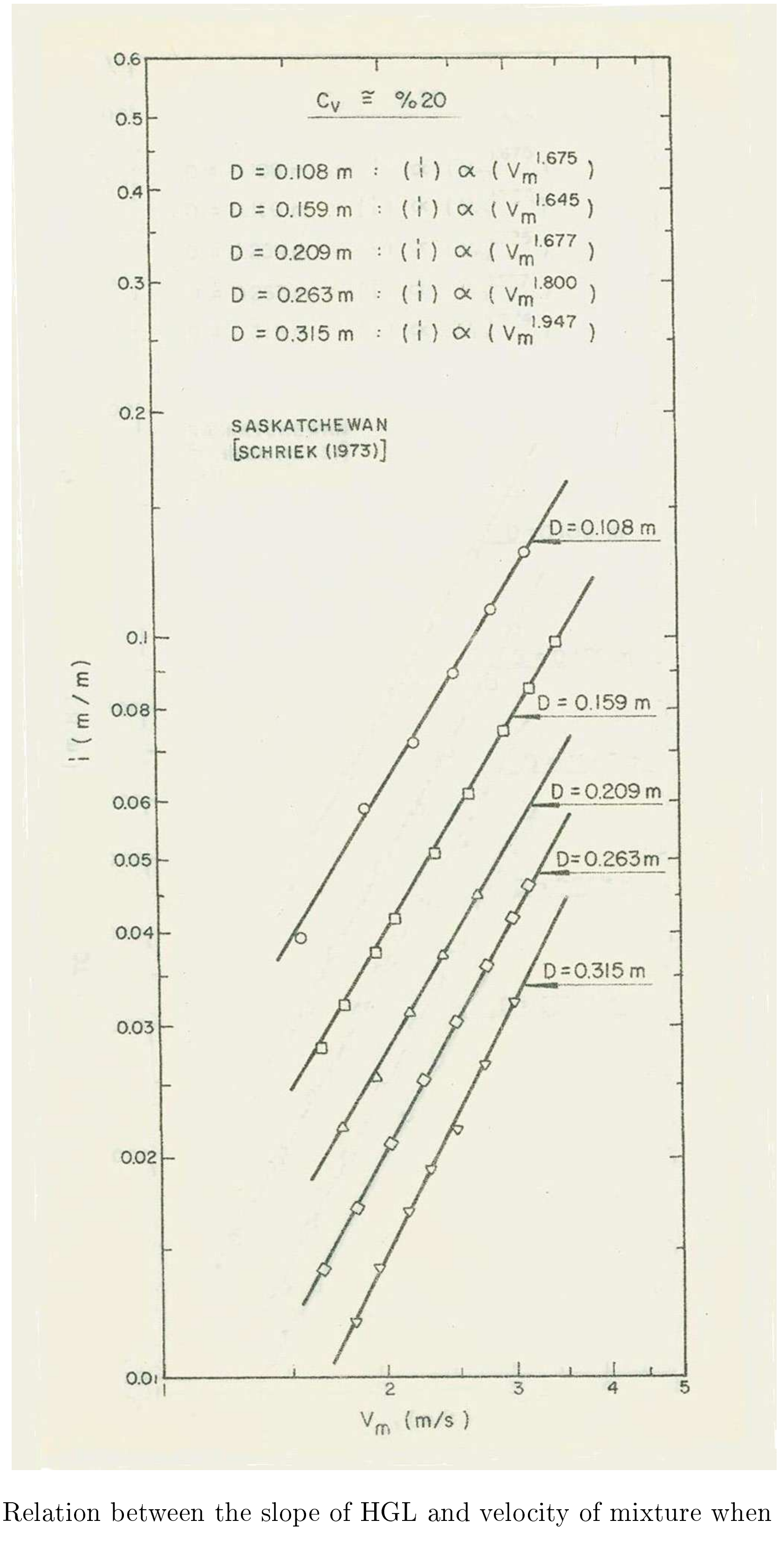

Figure 3.2: Relation between the slope of HGL and velocity of mixture when C_V=0.20 [2]

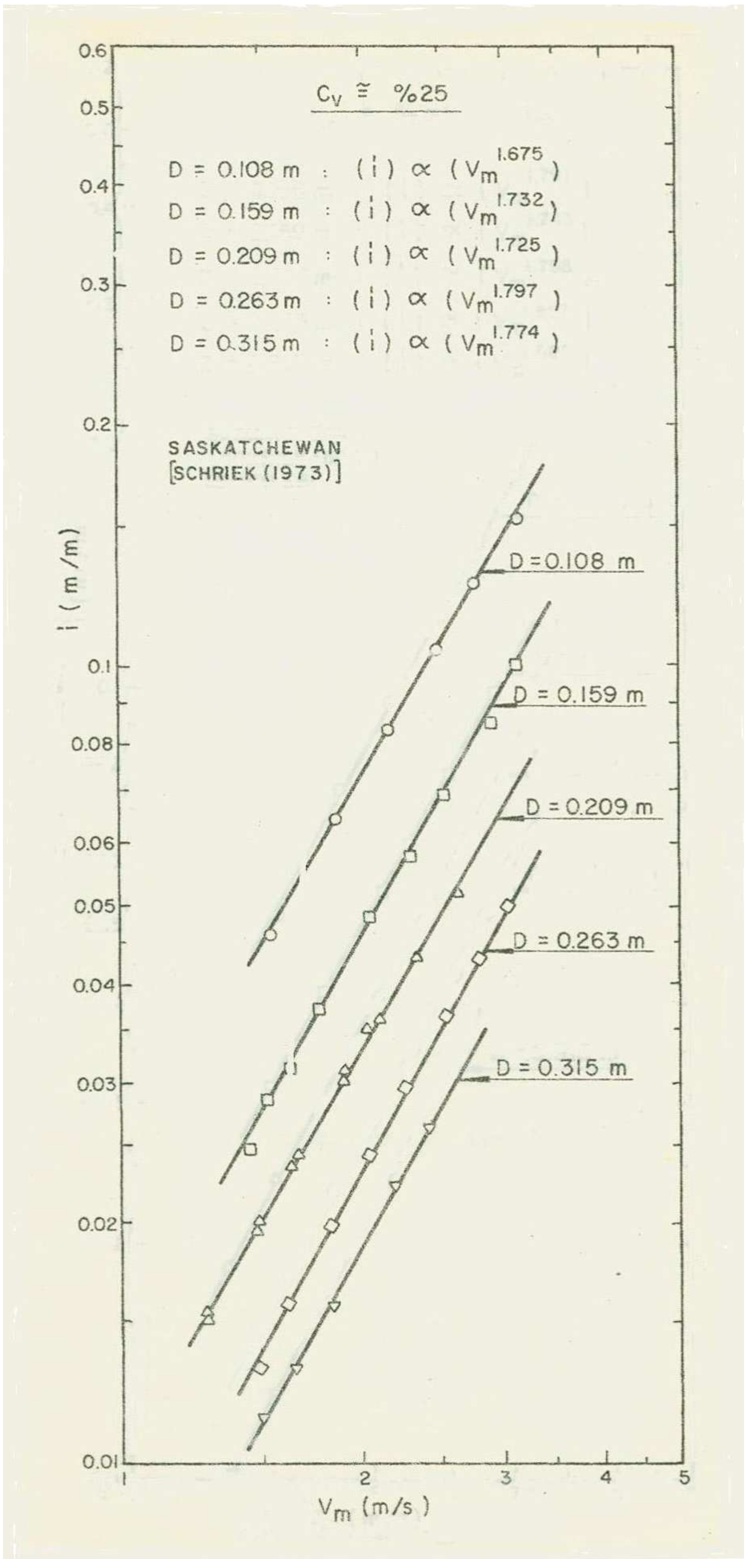

Figure 3.3: Relation between the slope of HGL and velocity of mixture when C_V=0.25 [2]

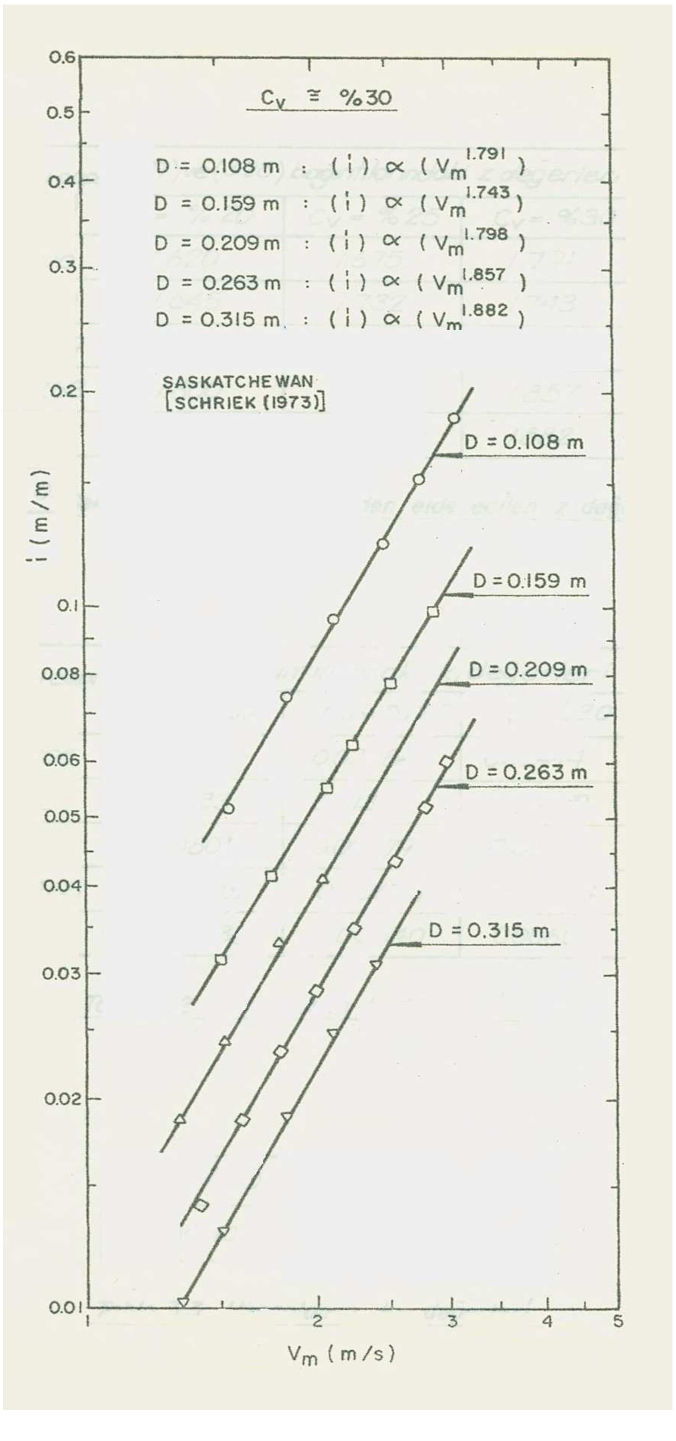

Figure 3.4: Relation between the slope of HGL and velocity of mixture when C_V=0.30 [2]

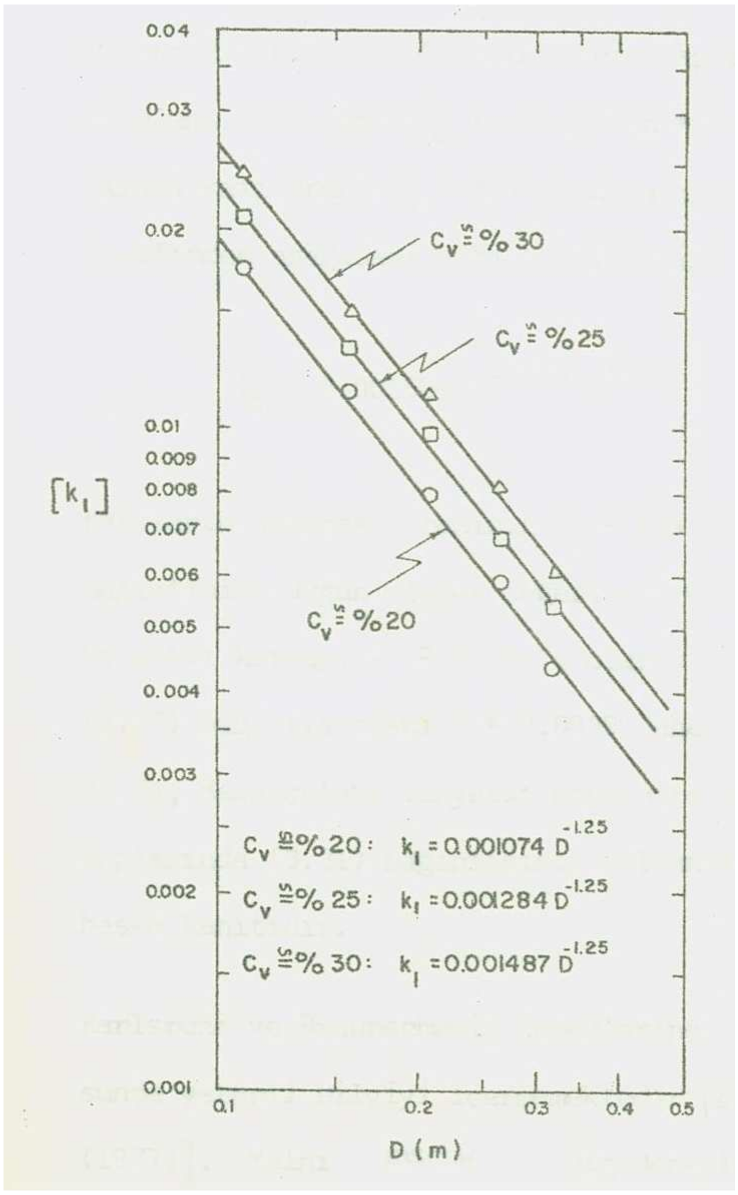

Figure 3.5: k_1 versus D for various C_V values [2]

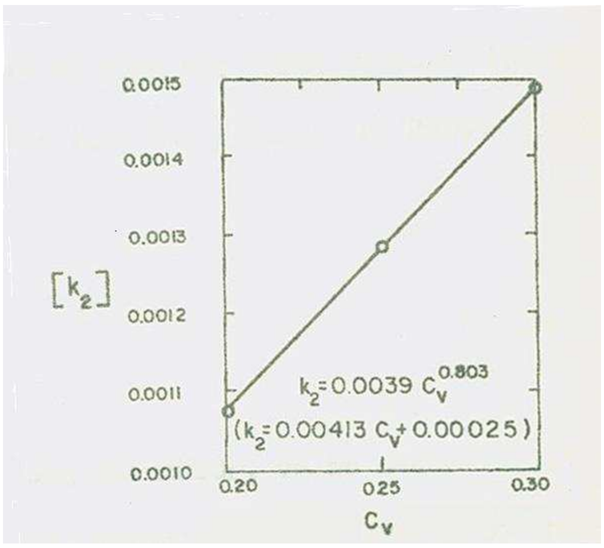

Figure 3.6: k_2 versus C_V[2]

Also another suggestion on head loss calculation has been made by Thomas, 1976 [14], for the iron ore materials having average particle diameters, $d_{50} = 40\ microns$ and mixtures having concentration by volume of, $C_V = 0.24$. The equation is shown below.

$$i = 0.001508 D^{-1.18} V_m^{1.77} \tag{3.6}$$

Taking C_V as 0.24, to compare Equation 3.5 with Equation 3.6, Equation 3.5 becomes

$$i = 0.001239 D^{-1.25} V_m^{1.77} \tag{3.7}$$

To see the differences of two equations more clearly Figure 3.7 is drawn. The agreement between the predictions of these two equations warrants to use of Equation 3.5.

Then the final state of head loss equation becomes by combining Equation 3.1 and Equation 3.5,

$$H = 0.0039 C_V^{0.803} D^{-1.25} V_m^{1.77} L \tag{3.8}$$

3.1.3 Critical Mixture Velocity

As stated in [1], the relation between critical mixture velocity (V_{cr}), particle diameter (d), concentration by weight (C_W), specific gravity (s_s) and diameter (D) of the pipe can be indicated as below,

$$V_{cr} = f(C_W) d^p s_s^q D^r \tag{3.9}$$

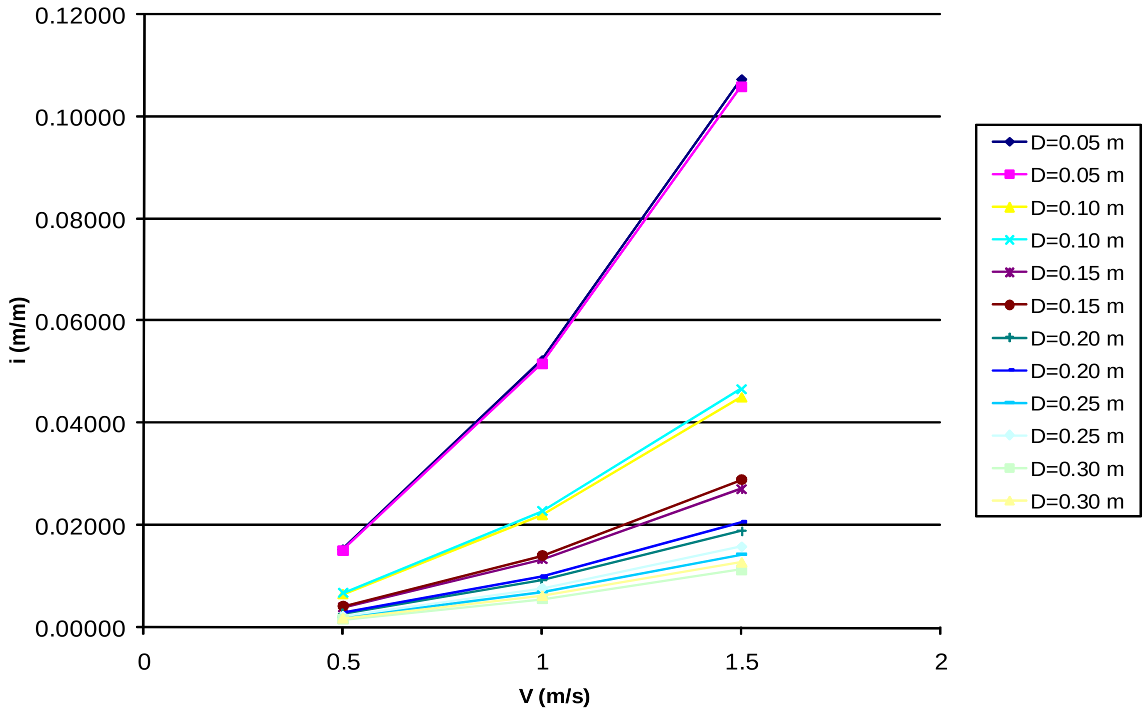

Figure 3.7: Comparison of Equations 3.6 and 3.7

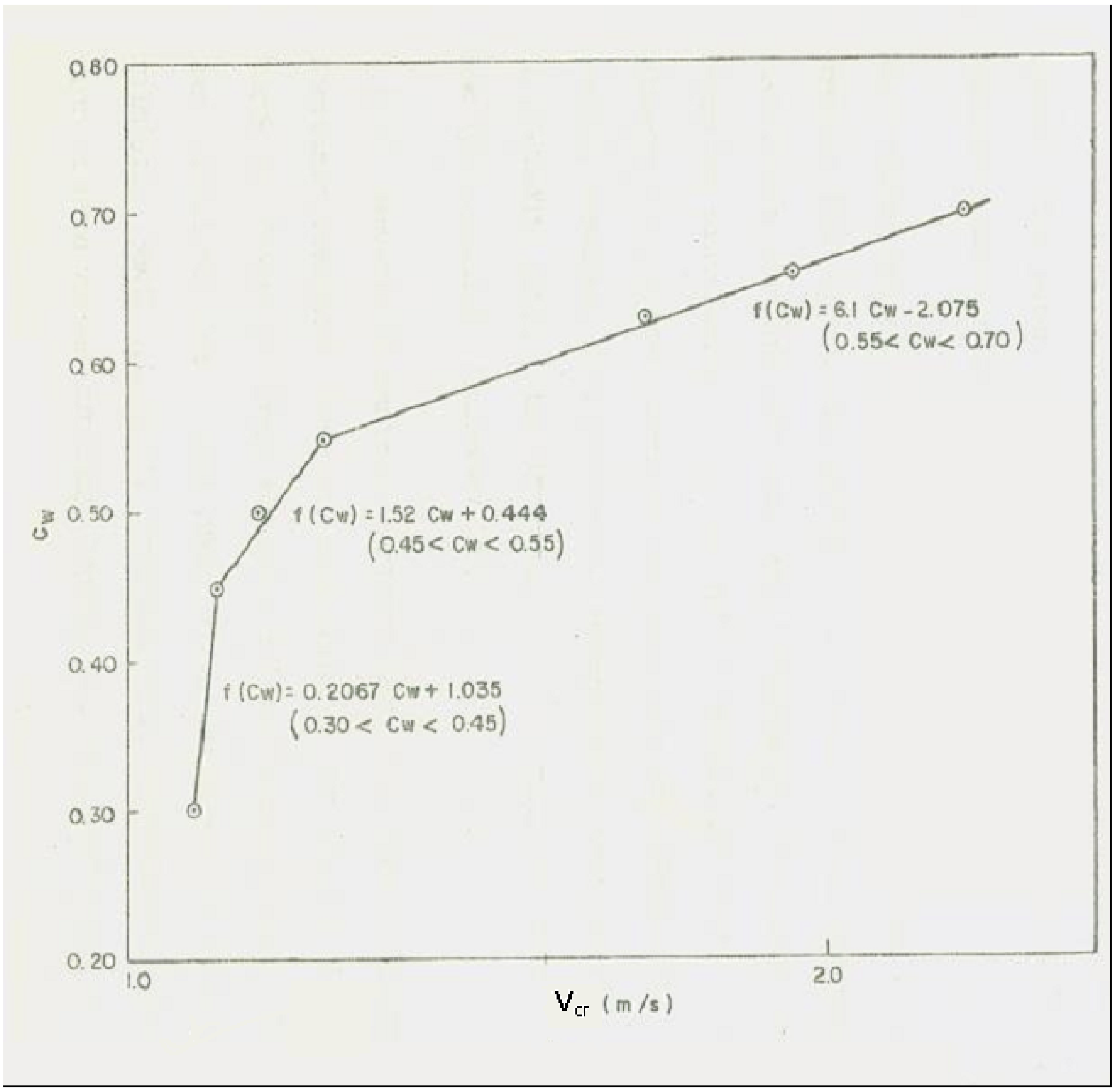

Figure 3.8: Relation between Critical Velocity and Concentration by Weight [3]

The values of the exponents are stated as below in the light of the experiments done in Colorado and Saskatchewan Institutes, [3] and [2],

$p = 0.75$, $q = 0.50$ and $r = 0.50$.

To find the value of $f(C_W)$, the graph shown in Figure 3.8 is utilized. In this figure, the experimental findings of the study made in Colorado Institute to investigate the relation between C_W and V_{cr} is depicted.

The researchers made a dimensional analysis to obtain the following equation [1].

$$V_{cr} = f(CW)(\frac{d}{d_*})^{0.75}(\frac{s_s}{s_{s*}})^{0.5}(\frac{D}{D_*})^{0.5} \tag{3.10}$$

In the experiments, particle diameter, $d_* = 20\ microns$, specific gravity, $s_{s*} = 4.71$ and diameter, $D_* = 0.27\ m$ was used. Placing the values in Equation 3.10, the following equation was obtained.

$$V_{cr} = 2966.45\, f(C_W)\, d^{0.75} s_s^{0.5} D^{0.5} \tag{3.11}$$

Referring to Figure 3.8 and using Equation 3.12, the relation between $f(C_W)$ and C_w is obtained for the ranges of C_w tested.

For $0.30 \leq C_W < 0.45$,

$$f(C_W) = 0.2067\, C_W + 1.035$$

For $0.45 \leq C_W < 0.55$,

$$f(C_W) = 1.5200\, C_W + 0.444 \tag{3.12}$$

For $0.55 \leq C_W \leq 0.70$,

$$f(C_W) = 6.1000\, C_W - 2.075$$

In Figure 3.8, there exists no information for $C_W < 0.30$, which is necessary for the present study. Taking the asymptotic behavior as depicted in the figure into account for $C_W < 0.30$, $f(C_W)$ is set equal to 1.097.

3.1.4 Cost Computations

Head loss of the whole system is calculated using Equation 3.8. Then the required pump power to supply the energy to overcome this loss can be found from;

$$P = \frac{\gamma Q H_p}{\eta} \tag{3.13}$$

where, P is the pump power, γ is the unit weight of mixture, Q is the discharge and H_p is the head of the pump. Equating the pump head to the head loss, Equation 3.13 can be written as,

$$P = \frac{\rho_m Q_m H}{101.94\eta} \tag{3.14}$$

where ρ_m (density of mixture) is in t/m^3, H (head loss) is in m and P results in kW. In this study, η (pump efficiency) is taken as 100% and Q_m (discharge of mixture) is in m^3/s. ρ_m is found by,

$$\rho_m = \rho_w(C_V\, s_s + (1 - C_V)) \tag{3.15}$$

where, ρ_w is density of mixture, C_V is concentration by volume, and s_s is specific gravity of solid material.

In Equation 3.14, H is found from Equation 3.8 and Q_m equals to V_{cr} (velocity) times A (area), where V_{cr} is obtained from Equation 3.11.

The energy supplied for 1-year time period is then found, assuming that the pump is in operation for 90% of the time, as follows;

$$E = 7884P \tag{3.16}$$

where, E is in kWh.

Finally, energy cost, $C1$ is computed by multiplying the unit price of energy with energy spent for one year.

$$C1 = CE \times E \tag{3.17}$$

where CE is the unit price of energy and taken as 0.1 $\$/kWh$ The electricity unit price for industrial facilities, announced by Turkish Energy Market Regularity Authority in 2008 as 0.172 TL/kWh, approximately 0.1 $\$/kWh$ and $C1$ results in \$.

Second component of the cost is the pipe cost, $C2$, which depends on the length and the diameter of the pipe. [12] suggests to use the following formula for pipe cost in pipelines.

$$C2 = 210.89D^{1.3744}L \tag{3.18}$$

where, $C2$ is in \$, D is in m and L is in m.

3.2 Definition of the Optimization Problem

3.2.1 Objective Function and Constraints

In the optimization problem part, it is decided to construct a problem of transporting iron ore from multi production points to multi consumption points. To develop this problem, m production points (mine pits) and n consumption points (factories) system is considered. The aim of the problem is to minimize the cost of all system. The decision variables become the pipe diameters and the concentrations of solid material that will be carried by each pipeline. The total cost of the system is the sum of the energy cost and the pipe cost. Thus, the cost of the system becomes as,

$$C = \sum_{i=1}^{m}\sum_{j=1}^{n} C1_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{n} C2_{ij} \tag{3.19}$$

where, $C1_{ij}$ stands for the energy cost associated with the head loss of pipe connecting i^{th} production point to j^{th} consumption point. Similarly, $C2_{ij}$ is the pipe cost of the corresponding pipe between i and j. $C1$ and $C2$ are calculated from Equation 3.17 and Equation 3.18, respectively. Thus, C becomes the total cost of the system.

The constraints of the problem are the capacities of the production points and the demands of the consumption points.

Thus, if WS_{ij} stands for the weight of the materials transported between i^{th} production point and j^{th}consumption point, the total weight of the ore produced on i^{th} production point (p_i) becomes;

$$p_i = \sum_{j=1}^{n} W_{ij} \qquad for\, i = 1, ..., m \tag{3.20}$$

Similarly for consumption points, the total weight of the ore used on j^{th} consumption point (c_j) becomes;

$$c_j = \sum_{i=1}^{m} W_{ij} \qquad for\, j = 1, ..., n \tag{3.21}$$

The weight of the transported material is calculated by;

$$W_{ij} = C_{Vij}\,(\rho_w\, s_s)\,\frac{\pi\, D_{ij}^2}{4}\,(V_m)_{ij} \qquad for\, i = 1, ..., m\; j = 1, ..., n \tag{3.22}$$

where, V_m is taken as V_{cr},also CV_{ij} can be written as;

$$C_{Vij} = \frac{C_{Wij}}{C_{Wij} + s_s\,(1 - C_{Wij})} \qquad for\, i = 1, ..., m\; j = 1, ..., n \tag{3.23}$$

It can be seen that, W_{ij} is calculated as function of D_{ij} and C_{Wij}, which are the decision variables of the optimization problem.

There exists bound constraints on D_{ij} and C_{Wij} as;

$$D_{min} \le D_{ij} \le D_{max} \qquad for\, i = 1, ..., m\; j = 1, ..., n \tag{3.24}$$

$$C_{Wmin} \le C_{Wij} \le C_{Wmax} \qquad for\, i = 1, ..., m\; j = 1, ..., n \tag{3.25}$$

where, D_{min} and D_{max} are the minimum and maximum pipe diameters allowed to be used and C_{Wmin} and C_{Wmax} are the minimum and maximum values of the concentration by weight.

The constraints are mainly related to the amount of the transported material. They have been handled in two cases; the first one is if the total capacity of the production points is greater than the total demand of the consumption points. The second case is just the opposite of the first case, which is the case of demand of consumption points to be higher than the capacity of production points. Both cases are investigated below.

Initially, if the total capacity of production points is greater than the total capacity of consumption points (Equation 3.26), each consumption point can use the materials till the limit of its capacity, that is the demands of the consumption points are satisfied. In other words, the amount of used material on every consumption point must be equal to the demand of corresponding point (Equation 3.27).

$$\sum_{j=1}^{n} ccap_j < \sum_{i=1}^{m} pcap_i \tag{3.26}$$

$$c_j = ccap_j \qquad for\, j = 1, ..., n \tag{3.27}$$

where, $ccap_j$ stands for the demand capacity of j^{th} consumption point and $pcap_i$ stands for the production capacity of i^{th} production point.

While satisfying the total demand of the consumption points, any production point should not exceed its own capacity as given in;

$$p_i \leq pcap_i \qquad for\, i = 1, ..., m \tag{3.28}$$

Also, the weight of materials transported on each pipe (W_{ij}) must be non-negative.

$$W_{ij} \geqslant 0 \qquad for\, i = 1, ..., m\; j = 1, ..., n \tag{3.29}$$

Equation 3.29 is automatically satisfied as W_{ij} is calculated from Equation 3.22 and with the bound limits of D_{ij} and C_{Wij}, it will always be non-negative.

On the other hand, for the second case, when the total demand of the consumption points is higher than the total capacity of the production points (Equation 3.30), the demand can not be satisfied. The production points must work on their maximum level and every production point should reach its capacity (Equation 3.31).

$$\sum_{i=1}^{m} pcap_i < \sum_{j=1}^{n} ccap_j \tag{3.30}$$

$$p_i = pcap_i \qquad for\; i = 1, ..., m \tag{3.31}$$

Besides, any consumption point should not exceed its own capacity.

$$c_j \leq ccap_j \tag{3.32}$$

Also, similar to the first case, the weight of materials transported on each pipe (W_{ij}) must be non-negative (Equation 3.33).

$$W_{ij} \geq 0 \tag{3.33}$$

If the production capacity and consumption capacity are to be equal, both cases will be valid. Therefore, either one of the smaller sign on Equation 3.26 or Equation 3.30 can be smaller or equal to sign.

3.2.2 Fitness Function of the Problem

In 3.2.1, the optimization problem is formulated basically by formulating the objective function and the constraints. The formulated optimization problem will be expressed in such a way that it will be possible to solve the problem using Genetic Algorithms (GA). It is known that, GA can only handle unconstrained optimization problems. The problem in hand is a constrained one. Hence, the constrained optimization problem must be converted to an unconstrained one. To achieve this, the penalty function method is used. In this method, the constraints are embedded into the objective function as penalty terms. Hence the objective function, that is to be solved by GA, now named as fitness function composed of the original objective function and penalty terms. Thus, the fitness function becomes;

$$F = C + P \tag{3.34}$$

where; C is the objective function (Equation 3.19), P is the penalty function. To find penalty function, the bracket penalty function method [15] is used in the study, which is shown for minimization problems as,.

$$P = R \sum_{l} [max(0, V_l)]^2 \tag{3.35}$$

where R is the penalty parameter which is a numerical value that must be assigned, V_l is the violation of bound constraint l. Penalty parameter for the problem must be chosen as a very big number. The value is chosen not to have any penalty violation for the results obtained, i.e. the constraints would be totally satisfied.

For example, for Equation 3.28, the corresponding numerical value of the violation of the constraint is $(p_i - pcap_i)$. Hence, the penalty term;

$$P = R[max(0, p_i - pcap_i)]^2 \tag{3.36}$$

If there exists no violation, if $p_i \leq pcap_i$, the resulting penalty term will be zero. If not, if $p_i \geq pcap_i$, the violation term will be positive and will be included into the objective function as a penalty term.

It is really very hard to satisfy the constraints like Equation 3.27. As c_j is the capacity of the consumption point j and calculated as the summation of weight of the substances coming to consumption point through each production point, exactly having c_j to be equal to the capacity of the consumption point $ccap_j$ is almost impossible. Hence, there will be no feasible solution to the optimization problem in consideration. If this constraint is relaxed to $c_j \leq ccap_j$, then as the objection is to minimize the cost, there will be no transportation of material between the production and consumption points which will make the objective function value to be zero. To force the transportation, Equation 3.27 is relaxed to;

$$\alpha\, ccap_j \leq c_j \leq ccap_j \qquad for\; j = 1, ..., n \tag{3.37}$$

α in this study is taken as 0.99. By the help of above constraint, the consumption point will work at almost its capacity.

Similarly, Equation 3.31 is relaxed to;

$$\alpha\, pcap_i \leq p_i \leq pcap_i \qquad for\; i = 1, ..., m \tag{3.38}$$

Hence, the fitness function of the optimization problem to be solved using GA will be composed of the objective function (Equation 3.19) and the penalty term associated with Equation 3.28 and Equation 3.37 for case I and Equation 3.32 and Equation 3.38 for case II. The bounds on D_{ij} and CW_{ij} are inputs to the GA code, hence, they are automatically satisfied.

3.3 Computer Application

After constructing the problem, a computer code is required to apply the genetic algorithms process. By this need, a commercially available software written in C language by Prof. Kalyonmey Deb from KANGAL (Kanpur Genetic Algorithms Laboratory) in 1995, is used. The code is written to solve the input optimization problem by using genetic algorithms method. Therefore, the constructed optimization problem is adapted to the C language and inserted to the code. The inserted part is given in the Appendix. The details are explained below.

3.3.1 Code Modification

Basically, the related equations of the problem are inserted to the GA code by adapting them into the C language. Also, some alternations were implemented. For instance, some modifications related to the limits of decision variables were done. The limits of C_W are used as 0 to 0.7 which is obtained from Colorado Researches. First of all, according to the code, the resultant CW were taken any real number within these limits. Then it is taught that using any real number as the solution would not be practical, so an extra limit is put by using the if function of C. By adding the algorithm given below, C_W will be chosen from a set of {0.01, 0.02, 0.03,......, 0.70}.

```
if (CW<0.005){
CW=0;}
else if ((0.005≤CW<0.015)){
CW=0.01;}
else if ((0.015≤CW<0.025)){
CW=0.02;}
Goes like this till upper limit
```

Similarly, a modification is done to get diameter values from commercially available diameters set. Limits of the diameter values are firstly decided as 0.1 and 1.0. The lower limit is given to use all pipes. However, in the analysis of results part it is shown that not to use some pipes would give better results. Then, the limits of the diameters is altered to 0 and 1.0. Similar algorithm is developed as CW, which is shown below,

```
if (D<0.11){
D=0.1; }
else if ((0.11≤D<0.135)) {
```

```
D=0.12; }
else if ((0.135≤D<0.175)){
D=0.15; }
else if ((0.175≤D<0.225)){
D=0.2; }
```

Goes like this till upper limit

Also, there exists max function on penalty function (Equation 3.35). when modifying the code; this mathematical operator has been removed and a new algorithm by using the if statement of C language has been established for its place. The algorithm works as, if the violation is greater than zero, it will be valid, else it will not. An example of the algorithm is given below.

```
for (i=0; i < pn; i++){
if (p[i] > pcap[i]) fitness = fitness + R*(p[i] - pcap[i])*(p[i] - pcap[i]) ; }
```

where, pn is the number of production points.

Also for the same case shown above, penalty functions for utilization points are shown below;

```
for (j=0; j < cn; j++){
if (c[j] > ccap[j]) fitness = fitness + R * (c[j] - ccap[j]) * (c[j] - ccap[j]) ;
if (c[j] < 0.99 * ccap[j]) fitness = fitness + R * (0.99 * ccap[j] - c[j]) * (0.99 * ccap[j] - c[j]) ; }
```

where, cn is the number of consumption points.

CHAPTER 4

APPLICATIONS AND RESULTS

The method developed in the study is explained briefly in Chapter 3. In Chapter 4, initially, applications for a 1-pipe system and for a model are presented. Then, the results and the analysis of the results are presented in this chapter.

4.1 Application and Results for 1-Pipe System

Initially, the code is run for 1-pipe system. A system, transporting materials from 1 production point to 1 utilization point is handled. The length between two points is taken as 400 km. Firstly, the transported material (W) is taken as 20 million tons/year. Then the code is run for weights of 10 and 15 million tons/year to see how optimal solution is changed by the weight of the transported material. The C code is adapted as diameters become the input of the problem and just concentrations become decision variables. The code is run several times for various diameters. The optimum concentration by weight values for every diameter value are obtained. Then all results are combined to get the optimal diameter and corresponding optimal concentration value. The results are shown in Figure 4.1. The change of two parts of the total cost, that are, the change of pipe costs and energy costs by diameter are also shown in the figure. The optimum values of diameter (D) and concentration by weight (CW) are obtained as 0.61 m and 0.446, respectively. As seen from Figure 4.1, pipe cost increases by increase in diameter as mentioned in Equation 3.18 and energy cost decreases by increase in diameter like explained in 3.1.4.

Also, the change of the optimum point with respect to the change of weight of the transported solid material is investigated. Then, the same problem is solved for weights of 10 and 15 million tons/year, respectively. The graph thus constructed is presented in Figure 4.2. It is seen from the graph that, when the amount of transported material increases, the cost of the system and also the optimal diameter increase.

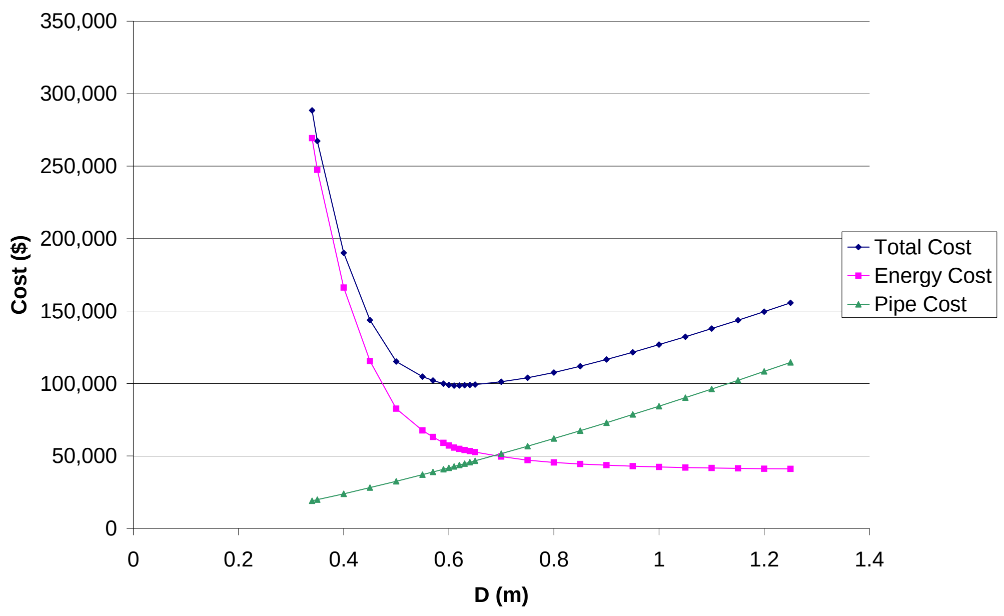

Figure 4.1: Cost versus Diameter for 1-Pipe-System

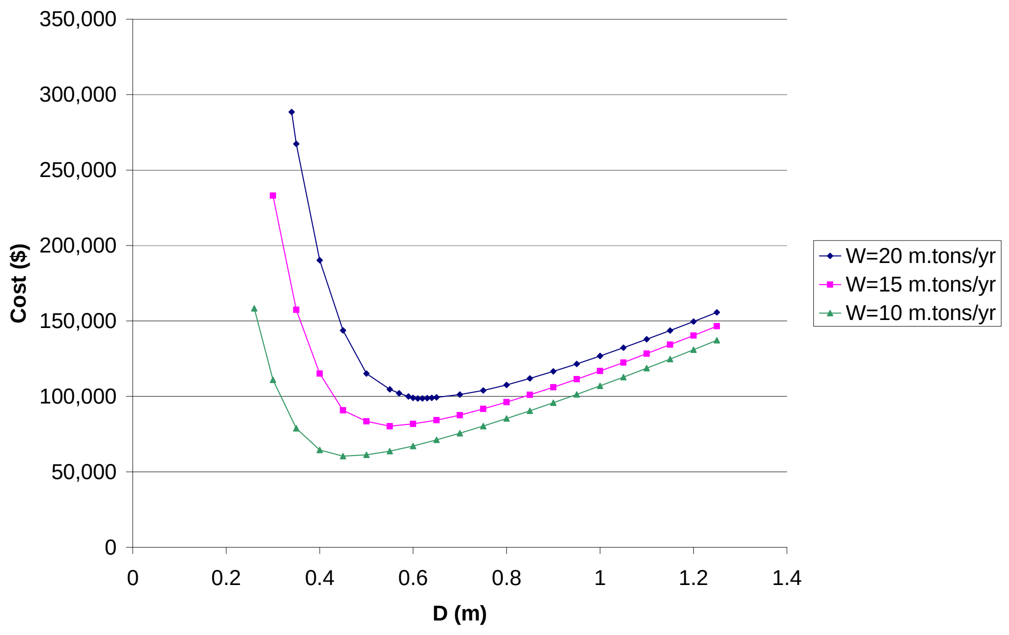

Figure 4.2: Cost versus Diameter for Different Weights of Material for 1-Pipe System

Additionally, another deduction from these runs are made about the concentration by weight(CW) and diameter relation. The optimal CW values for the diameter values shows an almost linear relation with the diameter values for CW>0.3. In Figure 4.3, the data points

Table 4.1: Distances Between Mine Pits and Steel Factories in kms.

	Hasançelebi	Avnik	Kozan
İskenderun	400	589	105
Samsun	583	901	988
Sivas	180	501	585

greater than CW=0.3 and smaller than CW=0.3 are separated, and the linear relation for points greater than CW=0.3 is obtained.

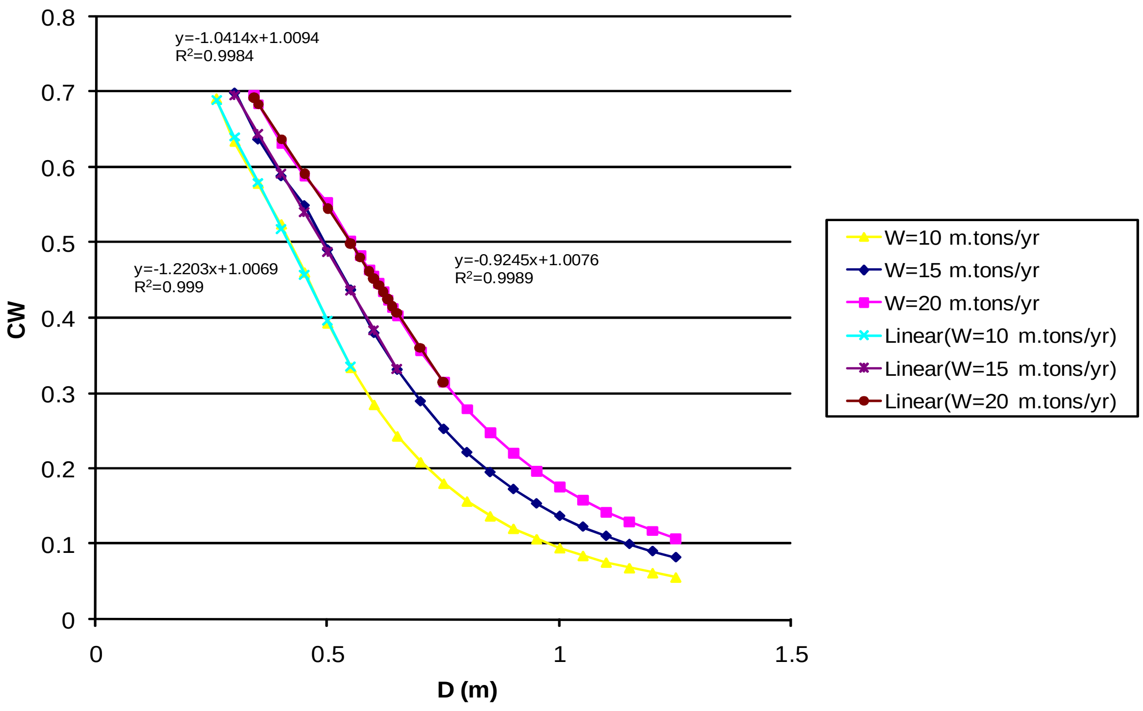

Figure 4.3: Relationship Between Concentration by Weight and Diameter in 1-Pipe-System

4.2 Model Application

The developed method is applied to an example problem with 3-inlet and 3-outlet points. This problem, which is also used as a case study in [5], is the optimization of transportation of Kozan, Avnik and Hasançelebi mine pits' iron ore to the steel factories located in İskenderun, Sivas and Samsun. The distances between mine pits to the steel factories, i.e. the lengths of the pipelines laid out, is given in Table 4.1. The capacities of the facilities are taken to be equal to the capacities used in the second case in [5], which are listed in Table 4.2.

Table 4.2: Capacities of the Facilities

Mine Pit	Production Capacity (10^6tons/yr)	Factory	Utilization Capacity (10^6tons/yr)
Hasançelebi	20	İskenderun	10
Avnik	10	Samsun	10
Kozan	5	Sivas	10

Thus, there occurs 9 pipelines to connect three production points to three utilization points. Then, 9 pipe diameters and 9 concentrations of solid materials will be the unknowns of the problem. Totally, there are 18 unknowns for the problem. The results of the optimization problem is given in Section 4.4 in detail.

4.3 Selection of the Genetic Algorithm Operators

After developing the optimization problem and adapting the code into the problem, the code has been run for several times for various combinations of genetic algorithms operators to obtain the optimal combinations of operators that will be used in the study.

First of all, generation number has been changed for various fixed population numbers ranging between 500 to 10000 and the results are shown in Figure 4.4. Other parameters are used as the default values of the code. It is seen from the figure that, increasing the generation number has a positive effect and decreases the fitness value. However, when the generation number exceeds 200, for all populations considered, there exists no improvement in the fitness value. Therefore, it is decided that taking generation number as 200 will be enough to get the optimum fitness value and also it will be more practical rather than dealing with higher generation numbers for the problem under consideration.

It can also be seen from Figure 4.4 that, the minimum fitness value is obtained when population number is equal to 9000. This can be seen more clearly, if the fitness versus population is drawn for each generation number used, which is shown in Figure 4.5. This figure shows the results of the same runs of Figure 4.4, having population range between 500 and 10000 and generation number range of 10 to 1000. Unfortunately, there exists no asymptotic behavior as in Figure 4.4, and the optimum results are obtained when population number equals to 9000. When the population number increases to 9500 slightly higher value of fitness value is obtained. Therefore it is decided to take population number as 9000 for

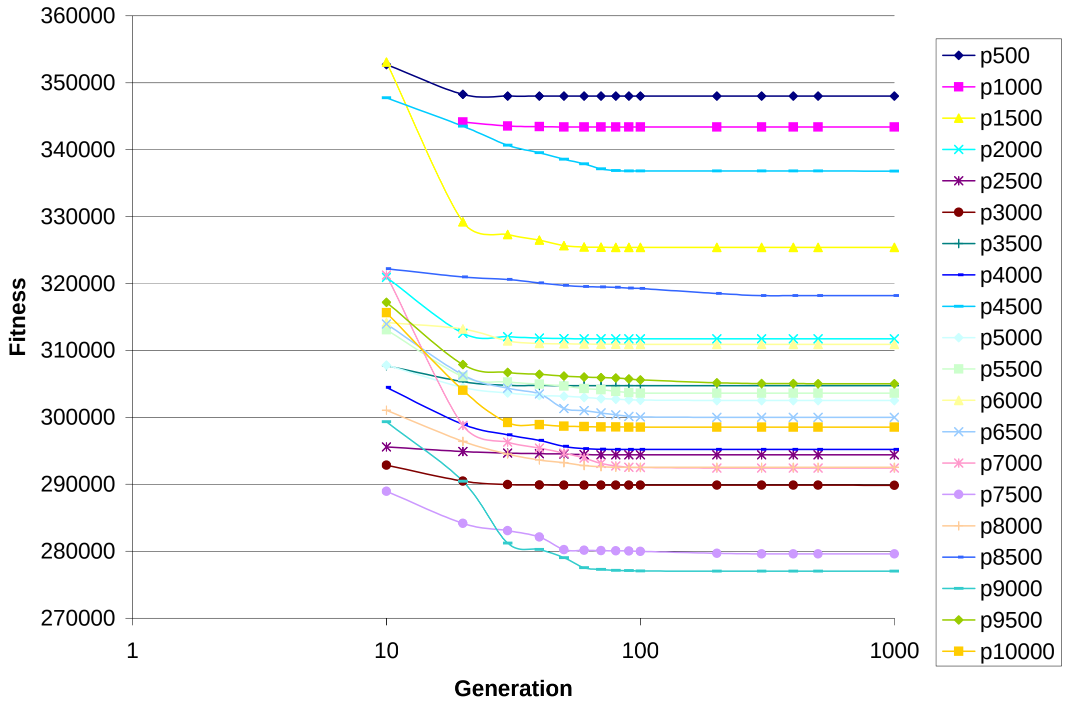

Figure 4.4: Effect of Generation Number on Fitness

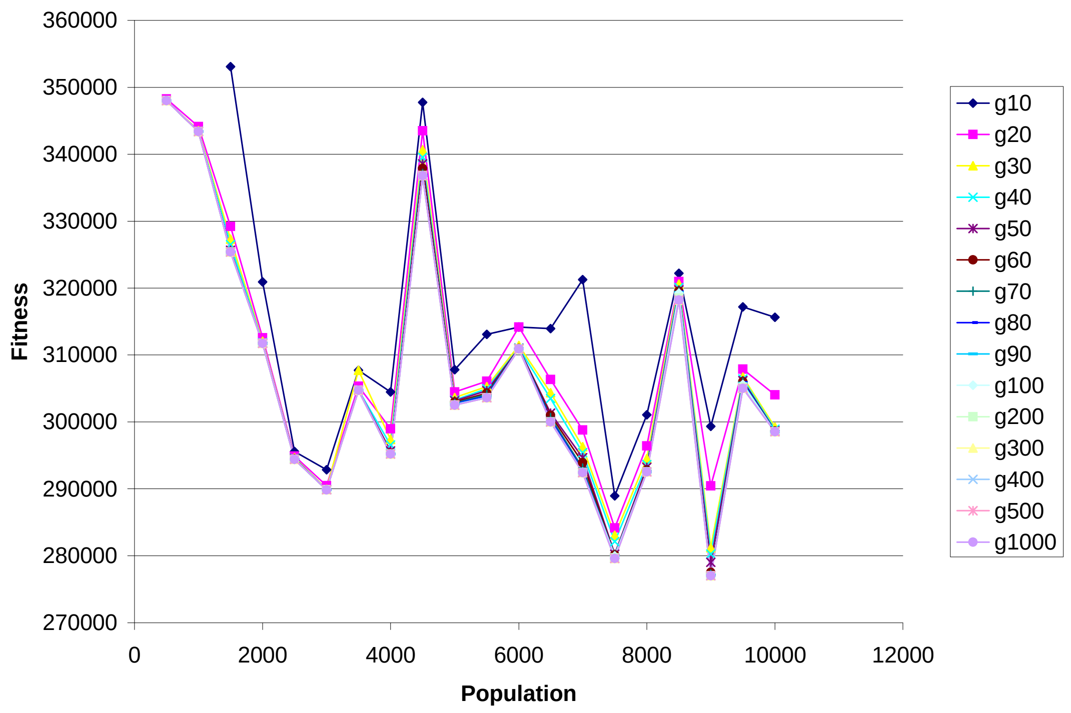

Figure 4.5: Effect of Population Number on Fitness

the further runs.

While optimum generation and population numbers are obtained, the code is run for the default values of the remaining parameters and also limitation to the values of CW has not been considered yet. Therefore, the minimum fitness value obtained from these runs are smaller than the following runs. The parameters found here is used for the following runs and after all parameters are found, these will be checked once more. It can be seen at the end of this Section.

Then, the effect of cross-over type on the fitness is observed. From this observation till the end of the study, the decision variables are limited as explained in Subsection 3.3.1.

As mentioned in Chapter 2, two cross-over types exists as SBX-η and BLX-α, and in the code used, η can take any number but α is defined as 0.5. Therefore, the runs are made for SBX-η and BLX-0.5. The runs are made for these two types for three different population numbers which give the smaller fitness values on previous runs. For generation number selection, since a relationship between generation number and fitness function is obtained in previous runs, trying different generation numbers is not required and the selected generation number (200) is used. Figure 4.6 is obtained, as a result of these runs. The effect of change of η on fitness value is observed from the figure. The runs made for same combinations for BLX-0.5, gives fitness values higher than the minimum of the one obtained by choosing sbx

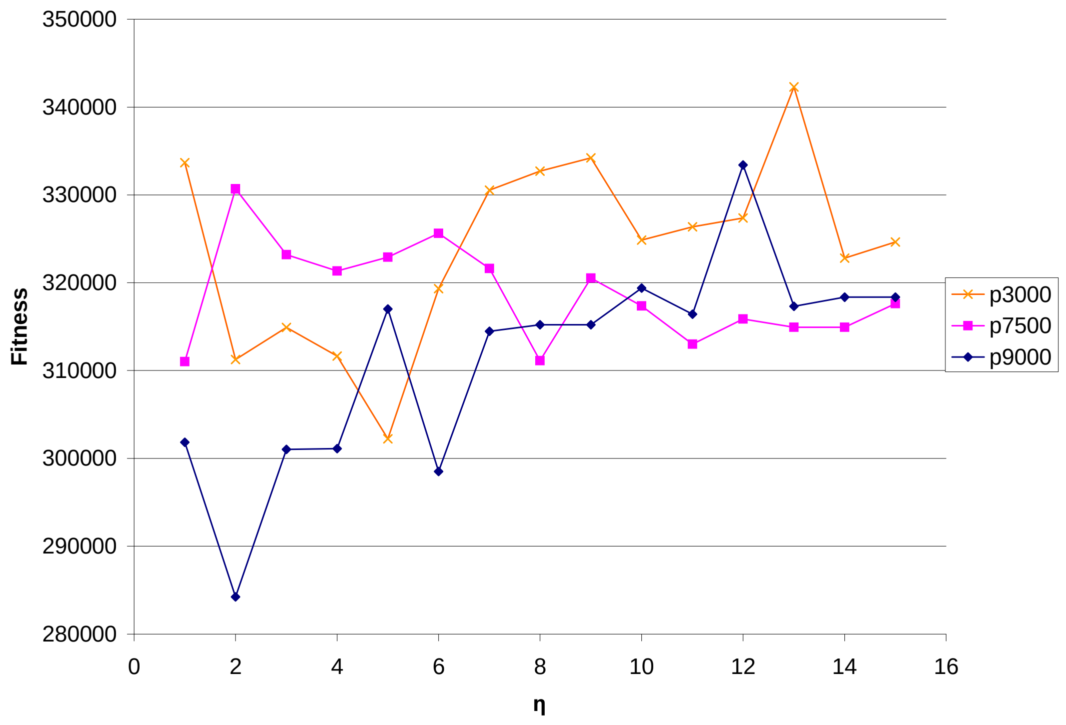

Figure 4.6: Effect of η on Fitness

type of cross-over. Therefore, blx type of cross-over excluded from the runs. SBX component of 2 will be in use as it gives the minimum fitness value. Also, the minimum is obtained when the population number is chosen as 9000, as can be seen from Figure 4.6.

Subsequently, the influence of cross-over rate on fitness is investigated. As seen from Figure 4.7, the minimum fitness value is obtained when cross-over ratio equals to 0.75. For this set of runs, population number, generation number and sbx component are chosen as 9000, 200 and 2, respectively.

Afterwards, the effect of mutation ratio on fitness value is investigated. For that purpose, by using the selected values for the operators observed previously, the code is run for various mutation ratio ranging from 0.01 to 0.15. Figure 4.8 is gathered from the study. It is seen from the figure that, for mutation number equals to 0.06, the fitness value is minimum. Therefore, this value is selected to be in use for the study.

As mentioned in Chapter 2, tournament selection operator is recommended for minimization problems. Therefore, it is selected as the selection operator of this problem. To decide on the tournament size, code runs are made to get the relation between tournament size and fitness function value. The resultant figure is presented in Figure 4.9. It is obtained from the figure that, for small values of tournament size, relatively small fitness values are obtained and the minimum is obtained when tournament size equals to 3.

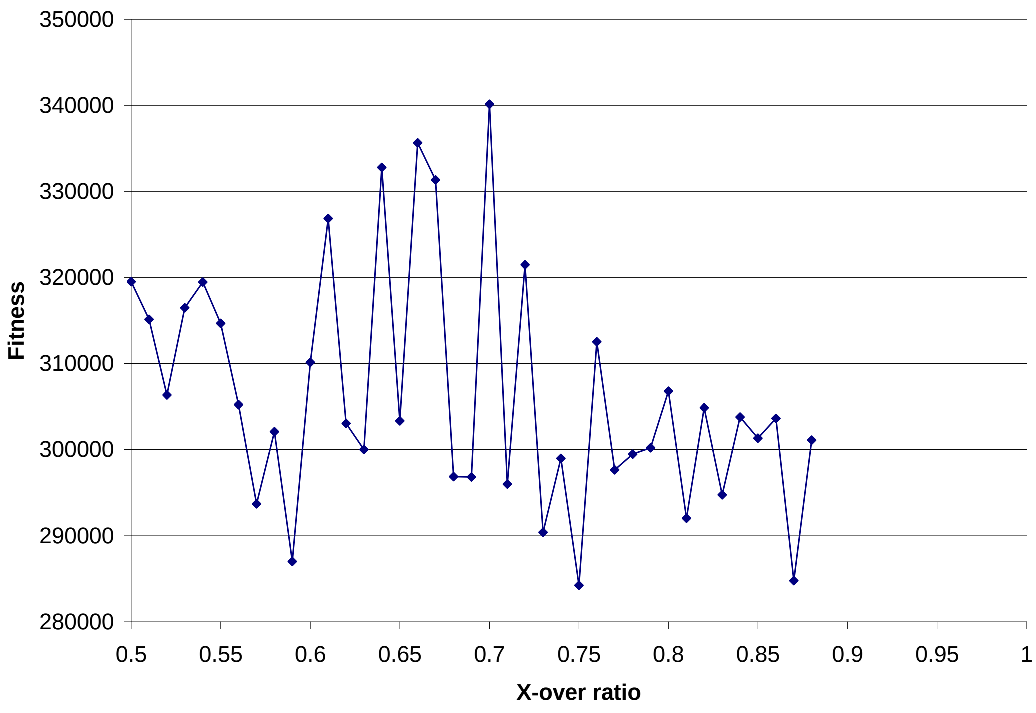

Figure 4.7: Effect of Crossover Ratio on Fitness

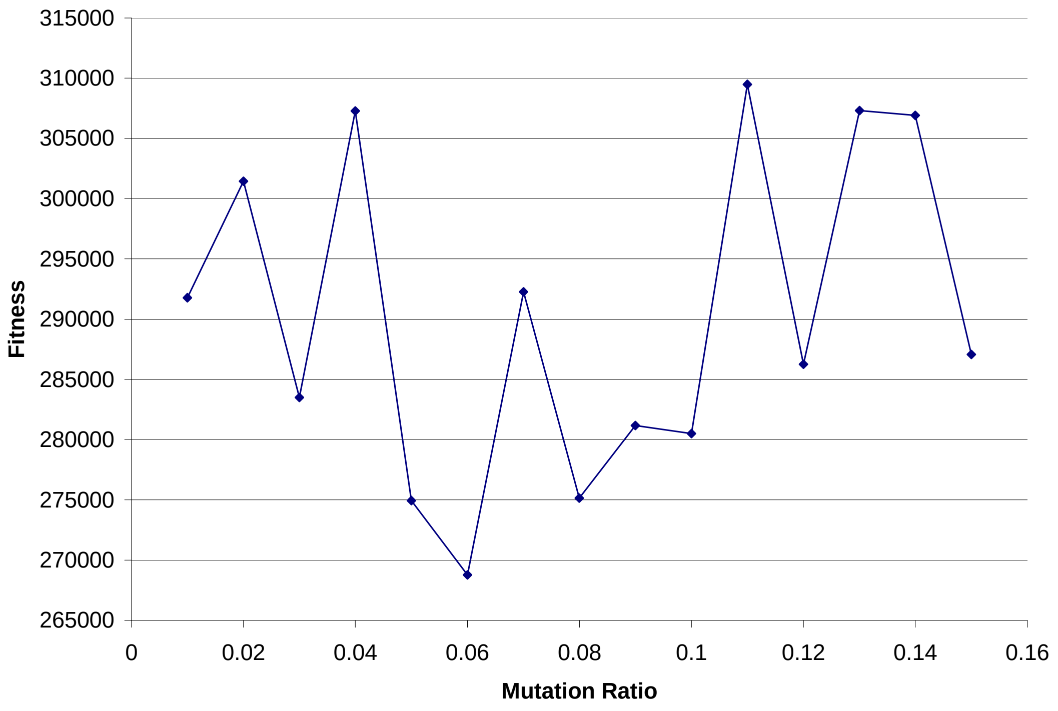

Figure 4.8: Effect of Mutation Ratio on Fitness

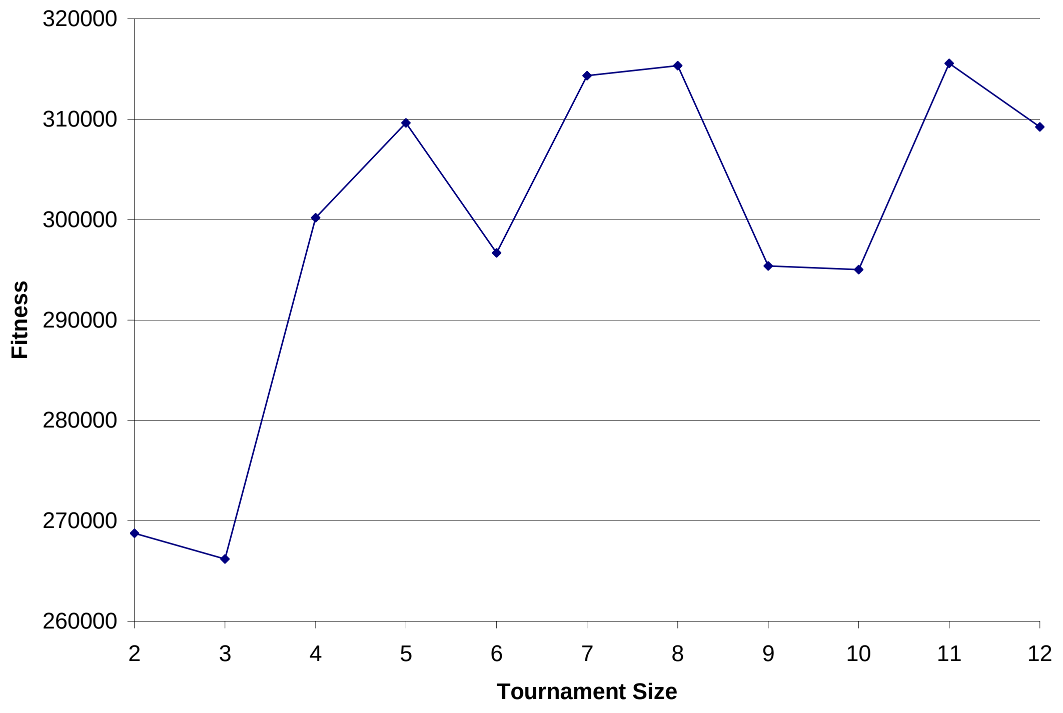

Figure 4.9: Effect of Tournament Size on Fitness

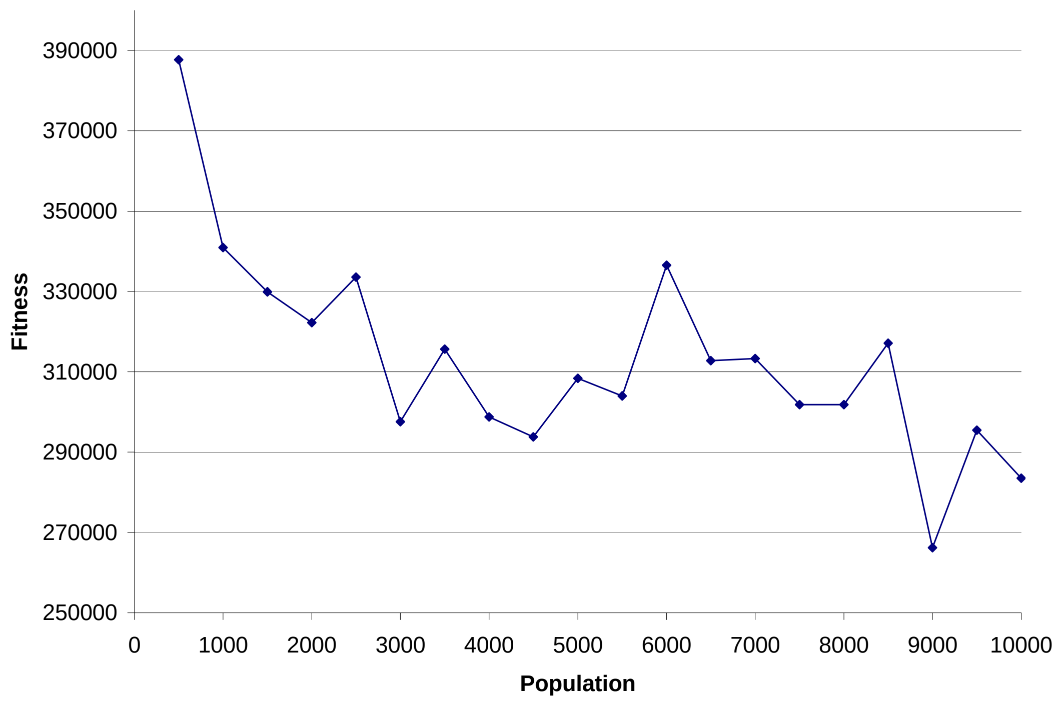

Figure 4.10: Check for the Selected Population Number

Finally, as a result of the studies to get the appropriate genetic algorithms operators and the numerical values of them, the operators are chosen as shown below;

Generation Number = 200,

Population Number = 9000,

Cross-over Type : sbx-2,

Cross-over Ratio = 0.75,

Mutation Ratio = 0.06,

Selection Operator : Tournament selection,

Tournament Size = 3.

Then finally, the check for the generation and population numbers are done. As stated above, generation number change does not have a big influence on fitness after generation number exceeds 200. Therefore, for the following runs generation number is fixed on 200 and the influence of population number change on fitness is observed, as can be seen on Figure 4.10. Again, for the Genetic Algorithm operators and the numerical values used, the optimum result is obtained when population is equal to 9000.

4.4 Results for the Model Application

The model application problem explained in Section 4.2 is solved by applying the parameters of the Genetic Algorithms found in the previous section. As explained in previously, decision variables are diameter and concentration by weight for the corresponding pipelines transporting between production and consumption points. The limits of the concentration by weight is taken as, 0 to 0.7, in the light of the information obtained from experiments explained in Subsection 3.1.3. The limits of the diameters are taken as 0.1 m to 1 m. The aim to take the lower limit as 0.1 m is to force to have all pipelines work. The upper limit is taken as 1 m, as it is thought that the exceeding values would not be practical to apply.

The penalty parameter is taken as $1{\times}10^9$ after trying several numbers. By using this number as penalty parameter, results having penalty term in fitness drops far away from the optimal solution. Therefore, by using this value, the probability of obtaining a feasible solution, constraints to be satisfied totally, would almost be 100%.

The result obtained from the solution of the problem is presented inTable 4.3. The cost is given in (1000 \$) to deal with small numbers in code. For this reason, a constant term in $Cost1$ and $Cost2$ functions separately divided to 1000.

Table 4.3: Results Obtained

(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*
Pipe	**D (m)**	**CW**	**WS(10^6 ton/year)**	**Cost1(1000\$)**	**Cost2(1000\$)**	**Total Cost(1000\$)**
p_{11}	0.5	0.34	7.979	22356	32537	54893
p_{12}	0.45	0.44	9.060	41386	41030	82416
p_{13}	0.15	0.62	1.587	7945	2799	10744
p_{21}	0.1	0	0.000	0	0	0
p_{22}	0.1	0.32	0.131	1470	8024	9494
p_{23}	0.45	0.41	8.102	31036	35259	66295
p_{31}	0.3	0.31	1.954	1702	4233	5935
p_{32}	0.2	0.31	0.709	6739	22811	29550
p_{33}	0.1	0.44	0.211	1674	5210	6884
Total			29.733	114308	151902	266210
Percentage				42.94	57.06	100.00

In Table 4.3, the first column elements, p_{ij}, stands for the pipeline between i^{th} production node and j^{th} consumption node. For example p_{11} is the pipeline laid out between first production point (Hasançelebi) and the first consumption point (İskenderun). The second and third columns in the table are the decision variables of the problem and these are the outputs of the Genetic Algorithms. Column 4 shows the weight of the material transported on each pipeline, which is a function of pipe diameter and concentration (Equation 3.22). Column 5 and column 6 shows the energy cost and pipe cost of each pipeline, respectively. In Chapter 3, energy cost is calculated using Equation 3.17 and the pipe cost is calculated using Equation 3.18. In column number 7, the total cost of the each pipelines are found by adding the elements of column number 5 and column number 6.

Thus, the weight of the material transported from the first production point (Hasançelebi) becomes the sum of WS values corresponding to the first three rows. Similarly, sum of the second three rows gives the weight of material transported from Avnik and the sum of the last three rows gives the weight of material transported from Kozan. The weight of material transported to the first consumption point (İskenderun) becomes, the sum of WS values of rows; 1, 4 and 7, as the pipelines connected to the first consumption point are, p_{11}, p_{21} and p_{31}. By the same manner for the remaining consumption points, the sum of second, fifth and eighth rows of WS is the weight of material transported to Samsun and the sum of the third, sixth and the ninth rows gives the weight of the material transported to Sivas.

The total capacities of the facilities were given in Table 4.2. As the total capacity of production points are greater than the total capacities of consumption points, case 1 (Equation 3.26) of the optimization problem is valid for this application. The numerical values of the weights of materials produced and consumed on each points are given in Table 4.4. When comparing Table 4.2 and Table 4.4, it is seen that, produced material weights of mine pits are less than the capacity of the corresponding pit. Also, the consumed material weights in the factories are within the limits of 0.99×capacity and the capacity of the corresponding factory. Therefore, all the constraints are satisfied, hence there is no violation. Therefore, no penalty value is added to the fitness and fitness is composed of only cost values and equals to 266210.

Also, it is seen from the percentages of the cost components that, pipe cost ($Cost2$) is slightly higher than the energy cost ($Cost1$).(57.06% to 42.94%)

It is seen from Table 4.3 that, in p_{21} nothing is transported as CW_{21} is 0, even though the optimum pipe diameter is found to be 0.1 m. Of course, it is meaningless to construct a pipeline, if nothing is transported. Hence, it is decided to lower the bound constraint of the pipe diameters to zero and be able to have no pipe connection between production and consumption points if it is not optimal. In order to see the fitness change and compare with the previous result, the problem is solved by changing the limits of diameter to 0 to 1. The results are presented in Table 4.5.

It is seen from the table that, pipes p_{23} and p_{32} are not in use. Therefore, the corre-

Table 4.4: Resultant Weights

Mine Pit	Weight of Produced Material (10^6tons/yr.)	Factory	Weight of Consumed Material (10^6tons/yr.)
Hasançelebi	18.626	İskenderun	9.933
Avnik	8.233	Samsun	9.900
Kozan	2.874	Sivas	9.900
Total	29.733	Total	29.733

Table 4.5: Results Obtained When the Lower Limit of Diameter is 0

(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*
Pipe	**D (m)**	**CW**	**WS(10^6 ton/year)**	**Cost1(1000$)**	**Cost2(1000$)**	**Total Cost(1000$)**
p_{11}	0.15	0.55	0.953	6003	6219	12222
p_{12}	0.45	0.42	8.413	37801	41030	78831
p_{13}	0.5	0.39	9.765	12736	14642	27378
p_{21}	0.35	0.51	6.627	40245	29345	69590
p_{22}	0.25	0.36	1.532	12614	28269	40883
p_{23}	0	0.07	0.000	0	0	0
p_{31}	0.3	0.36	2.417	2170	4233	6402
p_{32}	0	0.48	0.000	0	0	0
p_{33}	0.1	0.45	0.219	1751	5210	6961
Total			29.925	113320	128947	242267
Percentage				46.77	53.23	100.00

sponding pipe costs and energy costs are found to be 0. It should be noted that, in the second case better fitness value is obtained, as expected. Corresponding CW values, for pipes p_{23} and p_{32}, are just the values of the last population in Genetic Algorithms and they are meaningless values in the table.

If Table 4.5 is examined, it is seen that, the weight of the transported material in p_{33} is just $0.219{\times}10^6$ tons/yr. To construct pipeline between Kozan and Sivas to carry this amount of material might not be optimal, even though it is found to be the optimal result by the code. In real life application, construction of this pipeline should be reconsidered and alternatives must be considered.

Also, it is known that the system is solved for 1 year lifetime, so in cost computations, the initial cost and the annual cost are added directly. In this section to compare the results, the system is solved for 10-year and 50-year lifetime periods and the results are presented. Energy costs are taken as the same for the lifetime periods. The convergence of annual cost to present value is achieved by using the formula given below;

$$P = A\,[\frac{(1+i)^n - 1}{i\,(1+i)^n}] \tag{4.1}$$

where, n is the time in years, i is the interest rate, taken 10% yearly here, A is the annual cost and P is the present value of the cost.

In this study, as the energy cost of a year is assumed to be paid at the beginning of the year, Equation 4.1 becomes as,

$$P = A + A\,[\frac{(1+i)^{n-1} - 1}{i\,(1+i)^{n-1}}] \tag{4.2}$$

The results are shown in Table 4.6 and Table 4.7 for 10 and 50 years of life time, respectively. As seen from the results, diameter and concentration combination of 10-year and 50-year systems are almost same. The energy cost ($Cost1$) becomes more dominant in cost parts percentages, by increasing the life time period. As the total cost almost depends on one part of the cost by the increasing years, the algorithms starts to find out similar results for different life time periods.

Table 4.6: Results Obtained For Life Time = 10 years

(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*
Pipe	**D (m)**	**CW**	**WS(10^6 ton/year)**	**Cost1(1000$)**	**Cost2(1000$)**	**Total Cost(1000$)**
p_{11}	0.35	0.07	0.518	1813	19929	32183
p_{12}	0.55	0.33	9.705	38046	54060	311217
p_{13}	0.5	0.39	9.765	12736	14642	100727
p_{21}	0.55	0.32	9.294	36599	54617	301987
p_{22}	0	0.06	0.000	0	0	0
p_{23}	0	0.14	0.000	0	0	0
p_{31}	0.1	0.23	0.086	110	935	1682
p_{32}	0.1	0.42	0.196	2582	8799	26252
p_{33}	0.15	0.15	0.143	907	9096	15229
Total (Annual)			29.706	92794	162077	789276
Total (for 10 years)				627199	162077	789276
Percentage (for 10 years)				79.47	20.53	100.00

Table 4.7: Results Obtained For Life Time = 50 years

(1)	*(2)*	*(3)*	*(4)*	*(5)*	*(6)*	*(7)*
Pipe	**D (m)**	**CW**	**WS(10^6 ton/year)**	**Cost1(1000$)**	**Cost2(1000$)**	**Total Cost(1000$)**
p_{11}	0.35	0.07	0.518	1813	19929	39702
p_{12}	0.55	0.32	9.294	36226	54060	449149
p_{13}	0.5	0.39	9.765	12736	14642	153549
p_{21}	0.55	0.32	9.294	36599	54617	453771
p_{22}	0	0.05	0.000	0	0	0
p_{23}	0	0.19	0.000	0	0	0
p_{31}	0.1	0.23	0.086	110	935	2140
p_{32}	0.15	0.45	0.603	7028	15361	92013
p_{33}	0.12	0.23	0.135	908	6693	16597
Total (Annual)			29.694	95420	166238	1206920
Total (for 50 years)				1040682	166238	1206920
Percentage (for 50 years)				86.23	13.77	100.00

CHAPTER 5

CONCLUSION

The aim of this study was to optimize slurry pipeline systems with multi production and multi processing points. Due to high nonlinearity of the optimization problem, nonlinear optimization problem solving methods fail to converge to an optimal solution. In the present study, by using the Genetic Algorithms as the optimization method, a solution procedure is constructed. Genetic Algorithms is a previously tested and approved optimization method in many studies.

In the optimization problem, the minimization of the total cost of the system is aimed, by considering the weights of the transported material in each pipeline as constraints. The problem is solved by handling pipe diameters and concentration values as decision variables. First of all, this method is applied to a 1-pipe system to compare the results with the results obtained from previous studies. It is observed that, the results obtained in the present study show similarities with the ones obtained previously. Then the method is applied to a real life study. A system having 3 production points, 3 consumption points and 9 pipelines to connect these points, handled in the study. As the decision variables of the problem are pipe diameters and the concentration values of the transported material, in this application, there are 18 unknown parameters to be found. Although the number of the unknown parameters is high, the Genetic Algorithm achieved convergence to a solution in each run within the feasible region, as expected. The solution was improved further by identifying and implementing the optimal Genetic Algorithm parameters. After obtaining the results, the problem is solved once more by changing the limits of pipe diameters. The lower limit is reduced to zero, in the light of the information gathered from the first results. It is observed that, the results of the second case is better than the first case, as expected.

To conclude, in the present study, the optimum design of slurry pipelines is done by using Genetic Algorithms. The method developed in the study was shown to be an applicable method in the optimization of slurry pipelines with multi production and multi consumption points. For future research, cost function of the problem can be improved by inserting all possible cost parts into the objective function.

REFERENCES

[1] Ö. Yücel, A. S. Sevük, A. M. Ger, M. Z. Dogan, and M. Doruk, *Hasançelebi-İskenderun Arasında Boru Hattı ile Demir Cevheri Taşıması Teknik ve Mali Fizibilite Etüdü-Sonuç Raporu, ODTÜ Hidrolik Lab.*, Ulaştırma Bakanlıgı UKI 5.129, Ankara, 1978. x, 1, 2, 10, 11, 12, 13, 18, 20

[2] W. Schriek, L. Smith, D. Haas, and W. Husband, "Experimental studies on hydraulic transport of iron ore," 1973. x, 13, 14, 15, 16, 17, 18, 20

[3] F. Smith, "Pilot plant experiences with pipelines carrying mineral slurries," *First International Conference on the Hydraulic Transport of Solids in Pipes* , 1970. x, 20

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley Publishing Company, Inc., 1989. xii, 8, 9

[5] A. M. Ger and Ö. Yücel, "Çok kaynaklı cevher üretimi planlamasında boru hattı sistemleri ile taşımacılık yönünden optimizasyon," *5th National Conference on Operations Research and Industrial Engineering, Turkey* , 1979. 3, 31

[6] D. Goldberg and C. H. Kuo, "Genetic algorithms in pipeline design," *Journal of Computing in Civil Engineering* **1**(2), pp. 128–141, 1987. 3

[7] A. R. Simpson, G. C. Dandy, and L. J. Murphy, "Genetic algorithms compared to other techniques for pipe optimization," *Journal of Water Resources Planning and Management* **120**(4), pp. 423–443, 1994. 3

[8] M. Sharif and R. Wardlaw, "Multireservoir systems optimization using genetic algorithms: Case study," *Journal of Computing in Civil Engineering* **14**(4), pp. 255–263, 2000. 3

[9] R. Farmani, R. Abadia, and D. Savic, "Optimum design and management of pressurized branched irrigation networks," *Journal of Irrigation and Drainage Engineering* **133**(6), pp. 528–537, 2007. 3

[10] A. Popovic, A. Jankovic, D. Tosic, and V. Milutinovic, "Tutorial on genetic algorithms." `http://kondor.etf.rs/~vm/GenAlgo.ppt`, 01/12/2009. 7

[11] K. Deb and M. Goyal, "A combined genetic adaptive search for engineering design," *Journal of Computer Science and Informatics* , 1996. 7

[12] P. R. Bhave, "Optimal expansion of water distribution systems," *Journal of Environmental Engineering* **111**(2), pp. 177–197, 1985. 10, 22

[13] K. Wilson, G. Addie, A. Sellgren, and R. Cliff, *Slurry Transport Using Centrifugal Pumps*, Springer US, 2006. 11

[14] A. Thomas, "Scale-up methods for pipeline transport of slurries," *International Journal of Mineral Processing* (3), 1976. 18

[15] G. Li and L. W. Mays, "Differential dynamic programming for estuarine management," *Journal of Water Resources Planning and Management* **121**(6), pp. 455–462, 1995. 25

APPENDIX A

PART OF C CODE

```
float objective(x)
float *x;
{
int un = 3;
int tn = 3;
int p;
p = un * tn;

float C[p], C2[p], C3[p], C4[p];
float DEM[p], CV[p], CQM[p], CVM[p], FCM[p], CSI[p];
float L[9] = { 400000, 583000, 180000, 589000, 901000, 501000, 105000, 988000,
585000};
float C1[p],FCW[p];
float SS = 4.74;
float CWS[p], C5[p], WS[p];
float ucap[3] = { 634, 317, 158.5 } ;
float tcap[3] = { 317, 317, 317 };
float DEW = 1000.0;
float DP= 0.000045;
float CE= 0.0001;
float term3 = 0;
float g, h, penalty_coef;




        int i;
        for (i = 0; i < p; i++){

           if (0.11>x[i]){
           x[i]=0.1;
           }
           else if ((0.11<=x[i])&&(x[i]<0.135)) {
           x[i]=0.12;
           }
           else if ((0.135<=x[i])&&(x[i]<0.175)){
           x[i]=0.15;
           }
           else if ((0.175<=x[i])&&(x[i]<0.225)){
                x[i]=0.2;
           }
           else if ((0.225<=x[i])&&(x[i]<0.275)){
                x[i]=0.25;
           }
           else if ((0.275<=x[i])&&(x[i]<0.325)){
```

```
else if ((0.325<=x[i])&&(x[i]<0.375)){
                 x[i]=0.35;
            }
            else if ((0.375<=x[i])&&(x[i]<0.425)){
                 x[i]=0.4;
            }
            else if ((0.425<=x[i])&&(x[i]<0.475)){
                 x[i]=0.45;
            }
            else if ((0.475<=x[i])&&(x[i]<0.525)){
                 x[i]=0.5;
            }
            else if ((0.525<=x[i])&&(x[i]<0.575)){
                 x[i]=0.55;
            }
            else if ((0.575<=x[i])&&(x[i]<0.625)){
                 x[i]=0.6;
            }
            else if ((0.625<=x[i])&&(x[i]<0.675)){
                 x[i]=0.65;
            }
            else if ((0.675<=x[i])&&(x[i]<0.725)){
                 x[i]=0.7;
            }
            else if ((0.725<=x[i])&&(x[i]<0.775)){
                 x[i]=0.75;
            }
            else if ((0.775<=x[i])&&(x[i]<0.825)){
                 x[i]=0.8;
            }
            else if ((0.825<=x[i])&&(x[i]<0.875)){
                 x[i]=0.85;
            }
            else if ((0.875<=x[i])&&(x[i]<0.925)){
                 x[i]=0.9;
            }
            else if ((0.925<=x[i])&&(x[i]<0.975)){
                 x[i]=0.95;
            }
            else if ((0.925<=x[i])&&(x[i]<= 1)){
                 x[i]=1;
                 }
            if (0.005>x[i+9]){
            x[i+9]=0;
            }
            else if ((0.005<=x[i+9])&&(x[i+9]<0.015)) {
```

```
x[i+9]=0.01;
          }
          else if ((0.015<=x[i+9])&&(x[i+9]<0.025)) {
          x[i+9]=0.02;
          }
          else if ((0.025<=x[i+9])&&(x[i+9]<0.035)) {
          x[i+9]=0.03;
          }
          else if ((0.035<=x[i+9])&&(x[i+9]<0.045)) {
          x[i+9]=0.04;
          }
          else if ((0.045<=x[i+9])&&(x[i+9]<0.055)) {
          x[i+9]=0.05;
          }
          else if ((0.055<=x[i+9])&&(x[i+9]<0.065)) {
          x[i+9]=0.06;
          }
          else if ((0.065<=x[i+9])&&(x[i+9]<0.075)) {
          x[i+9]=0.07;
          }
          else if ((0.075<=x[i+9])&&(x[i+9]<0.085)) {
          x[i+9]=0.08;
          }
          else if ((0.085<=x[i+9])&&(x[i+9]<0.095)) {
          x[i+9]=0.09;
          }
          else if ((0.095<=x[i+9])&&(x[i+9]<0.105)) {
          x[i+9]=0.1;
          }
          else if ((0.105<=x[i+9])&&(x[i+9]<0.115)) {
          x[i+9]=0.11;
          }
          else if ((0.115<=x[i+9])&&(x[i+9]<0.125)) {
          x[i+9]=0.12;
          }
          else if ((0.125<=x[i+9])&&(x[i+9]<0.135)) {
          x[i+9]=0.13;
          }
          else if ((0.135<=x[i+9])&&(x[i+9]<0.145)) {
          x[i+9]=0.14;
          }
          else if ((0.145<=x[i+9])&&(x[i+9]<0.155)) {
          x[i+9]=0.15;
          }
          else if ((0.155<=x[i+9])&&(x[i+9]<0.165)) {
          x[i+9]=0.16;
```

```
}
          else if ((0.165<=x[i+9])&&(x[i+9]<0.175)) {
          x[i+9]=0.17;
          }
          else if ((0.175<=x[i+9])&&(x[i+9]<0.185)) {
          x[i+9]=0.18;
          }
          else if ((0.185<=x[i+9])&&(x[i+9]<0.195)) {
          x[i+9]=0.19;
          }
          else if ((0.195<=x[i+9])&&(x[i+9]<0.205)) {
          x[i+9]=0.2;
          }
          else if ((0.205<=x[i+9])&&(x[i+9]<0.215)) {
          x[i+9]=0.21;
          }
          else if ((0.215<=x[i+9])&&(x[i+9]<0.225)) {
          x[i+9]=0.22;
          }
          else if ((0.225<=x[i+9])&&(x[i+9]<0.235)) {
          x[i+9]=0.23;
          }
          else if ((0.235<=x[i+9])&&(x[i+9]<0.245)) {
          x[i+9]=0.24;
          }
          else if ((0.245<=x[i+9])&&(x[i+9]<0.255)) {
          x[i+9]=0.25;
          }
          else if ((0.255<=x[i+9])&&(x[i+9]<0.265)) {
          x[i+9]=0.26;
          }
          else if ((0.265<=x[i+9])&&(x[i+9]<0.275)) {
          x[i+9]=0.27;
          }
          else if ((0.275<=x[i+9])&&(x[i+9]<0.285)) {
          x[i+9]=0.28;
          }
          else if ((0.285<=x[i+9])&&(x[i+9]<0.295)) {
          x[i+9]=0.29;
          }
          else if ((0.295<=x[i+9])&&(x[i+9]<0.305)) {
          x[i+9]=0.30;
          }
          else if ((0.305<=x[i+9])&&(x[i+9]<0.315)) {
          x[i+9]=0.31;
          }
```

```
}
            else if ((0.315<=x[i+9])&&(x[i+9]<0.325)) {
            x[i+9]=0.32;
            }
            else if ((0.325<=x[i+9])&&(x[i+9]<0.335)) {
            x[i+9]=0.33;
            }
            else if ((0.335<=x[i+9])&&(x[i+9]<0.345)) {
            x[i+9]=0.34;
            }
            else if ((0.345<=x[i+9])&&(x[i+9]<0.355)) {
            x[i+9]=0.35;
            }
            else if ((0.355<=x[i+9])&&(x[i+9]<0.365)) {
            x[i+9]=0.36;
            }
            else if ((0.365<=x[i+9])&&(x[i+9]<0.375)) {
            x[i+9]=0.37;
            }
            else if ((0.375<=x[i+9])&&(x[i+9]<0.385)) {
            x[i+9]=0.38;
            }
            else if ((0.385<=x[i+9])&&(x[i+9]<0.395)) {
            x[i+9]=0.39;
            }
            else if ((0.395<=x[i+9])&&(x[i+9]<0.405)) {
            x[i+9]=0.40;
            }
            else if ((0.405<=x[i+9])&&(x[i+9]<0.415)) {
            x[i+9]=0.41;
            }
            else if ((0.415<=x[i+9])&&(x[i+9]<0.425)) {
            x[i+9]=0.42;
            }
            else if ((0.425<=x[i+9])&&(x[i+9]<0.435)) {
            x[i+9]=0.43;
            }
            else if ((0.435<=x[i+9])&&(x[i+9]<0.445)) {
            x[i+9]=0.44;
            }
            else if ((0.445<=x[i+9])&&(x[i+9]<0.455)) {
            x[i+9]=0.45;
            }
            else if ((0.455<=x[i+9])&&(x[i+9]<0.465)) {
            x[i+9]=0.46;
            }
```

```
else if ((0.465<=x[i+9])&&(x[i+9]<0.475)) {
            x[i+9]=0.47;
            }
            else if ((0.475<=x[i+9])&&(x[i+9]<0.485)) {
            x[i+9]=0.48;
            }
            else if ((0.485<=x[i+9])&&(x[i+9]<0.495)) {
            x[i+9]=0.49;
            }
            else if ((0.495<=x[i+9])&&(x[i+9]<0.505)) {
            x[i+9]=0.5;
            }
            else if ((0.505<=x[i+9])&&(x[i+9]<0.515)) {
            x[i+9]=0.51;
            }
            else if ((0.515<=x[i+9])&&(x[i+9]<0.525)) {
            x[i+9]=0.52;
            }
            else if ((0.525<=x[i+9])&&(x[i+9]<0.535)) {
            x[i+9]=0.53;
            }
            else if ((0.535<=x[i+9])&&(x[i+9]<0.545)) {
            x[i+9]=0.54;
            }
            else if ((0.545<=x[i+9])&&(x[i+9]<0.555)) {
            x[i+9]=0.55;
            }
            else if ((0.555<=x[i+9])&&(x[i+9]<0.565)) {
            x[i+9]=0.56;
            }
            else if ((0.565<=x[i+9])&&(x[i+9]<0.575)) {
            x[i+9]=0.57;
            }
            else if ((0.575<=x[i+9])&&(x[i+9]<0.585)) {
            x[i+9]=0.58;
            }
            else if ((0.585<=x[i+9])&&(x[i+9]<0.595)) {
            x[i+9]=0.59;
            }
            else if ((0.595<=x[i+9])&&(x[i+9]<0.605)) {
            x[i+9]=0.6;
            }
            else if ((0.605<=x[i+9])&&(x[i+9]<0.615)) {
            x[i+9]=0.61;
            }
            else if ((0.615<=x[i+9])&&(x[i+9]<0.625)) {
```

```
x[i+9]=0.62;
            }
            else if ((0.625<=x[i+9])&&(x[i+9]<0.635)) {
            x[i+9]=0.63;
            }
            else if ((0.635<=x[i+9])&&(x[i+9]<0.645)) {
            x[i+9]=0.64;
            }
            else if ((0.645<=x[i+9])&&(x[i+9]<0.655)) {
            x[i+9]=0.65;
            }
            else if ((0.655<=x[i+9])&&(x[i+9]<0.665)) {
            x[i+9]=0.66;
            }
            else if ((0.665<=x[i+9])&&(x[i+9]<0.675)) {
            x[i+9]=0.67;
            }
            else if ((0.675<=x[i+9])&&(x[i+9]<0.685)) {
            x[i+9]=0.68;
            }
            else if ((0.685<=x[i+9])&&(x[i+9]<695)) {
            x[i+9]=0.69;
            }
            else if ((0.695<=x[i+9])&&(x[i+9]<0.7)) {
            x[i+9]=0.6975;
            }




         CV[i]=x[i+9]/(x[i+9]+SS*(1-x[i+9]));
         DEM[i]=DEW*(CV[i]*SS+(1-CV[i]));


               printf("L= %f\n", L[i]);*/
         if ((0.3<=x[i+9])&&(x[i+9]<0.45)){
                  FCW[i]=0.2067*x[i+9]+1.035;
         }
         else if ((0.45<=x[i+9])&&(x[i+9]<0.55)){
                   FCW[i]=1.52*x[i+9]+0.444;
         }
         else if ((0.55<=x[i+9])&&(x[i+9]<0.70)){
                    FCW[i]=6.1*x[i+9]-2.075;
         }
         else if (x[i+9]<0.3){
```

```
     FCW[i]=0.2067*0.3+1.035;
      }

if (x[i+9]<0.7){
            C2[i]=2.135;
      CVM[i]=2966.45*FCW[i]*(pow(DP,0.75))*(pow(SS,0.5));
}

else {
       C2[i]=1000;
              CVM[i]=1000;




if (x[i+9]<0.7){
            C5[i]=2.5;
}

else {
              C5[i]=1000;
}

CWS[i]=CV[i]*DEW*SS*3.14159*CVM[i]/4;
CSI[i]=0.0039*(pow(CV[i],0.803));
CQM[i]=CVM[i]*3.14159/4;
C1[i]= CE*365*24*DEM[i]*CQM[i]*CSI[i]*(pow(CVM[i],1.77))*L[i]/101.94;
C3[i]= 0.21089*L[i];



C4[i]= 1.3744;
C[i]=C1[i]*(pow(x[i],C2[i]))+C3[i]*(pow(x[i],C4[i]));

WS[i]=CWS[i]*pow(x[i],C5[i]);
}
int y, z, a, b, s, r, e;
int cost = 0;
for (s=0; s<p; s++){
   cost = cost + C[s];
   }
float u[3], t[3];
u[0]= WS[0]+WS[1]+WS[2];
u[1]= WS[3]+WS[4]+WS[5];
u[2]= WS[6]+WS[7]+WS[8];
t[0]= WS[0]+WS[3]+WS[6];
```

```
t[1]= WS[1]+WS[4]+WS[7];
            t[2]= WS[2]+WS[5]+WS[8];
            int ucap_tot = 0;
            int tcap_tot = 0;


            for (r=0; r < un; r++){
                ucap_tot = ucap_tot + ucap[r];
                }

            for (e=0; e < tn; e++){
                tcap_tot = tcap_tot + tcap[e];
                }
        if (ucap_tot >= tcap_tot) {



#ifdef prob1
MINM  = 1;
term3 = cost;

   penalty_coef = 1000.0;
   for (a=0; a < un; a++){
   if (u[a] < 0.0) term3 = (term3 + penalty_coef * u[a] * u[a])/1000000 ;
   if (u[a] > ucap[a]) term3 = (term3 + penalty_coef * (u[a] - ucap[a]) * (u[a] -
ucap[a]))/1000000 ;
}
   for (b=0; b < tn; b++){
   if (t[b] > tcap[b]) term3 = (term3 + penalty_coef * (t[b] - tcap[b]) * (t[b] -
tcap[b]))/1000000 ;
   if (t[b] < 0.995 * tcap[b]) term3 = (term3 + penalty_coef * (0.995 * tcap[b] - t[b])  *
(0.995 * tcap[b] - t[b]))/1000000 ;
}

     return(term3);
#endif
   }



   if (tcap_tot > ucap_tot) {

#ifdef prob1
MINM  = 1;
term3 = cost;
```

```
  penalty_coef = 1000.0;
  for (y=0; y < tn; y++){
  if (t[y] < 0.0) term3 = (term3 + penalty_coef * t[y] * t[y])/1000000 ;
  if (t[y] > tcap[y]) term3 = (term3 + penalty_coef * (t[y] - tcap[y]) * (t[y] -
tcap[y]))/1000000 ;
}
  for (z=0; z < un; z++){
  if (u[z] > ucap[z]) term3 = (term3 + penalty_coef * (u[z] - ucap[z]) * (u[z] -
ucap[z]))/1000000 ;
  if (u[z] < 0.995 * ucap[z]) term3 = (term3 + penalty_coef * (0.995 * ucap[z] - u[z]) *
(0.995 * ucap[z] - u[z]))/1000000 ;
}
  return(term3);
#endif
}
}
```