NETWORK PARTITION FOR SWITCHED INDUSTRIAL ETHERNET USING COMBINED SEARCH HEURISTICS

A THESIS IN

Computer Science

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree

MASTERS OF SCIENCE

by

VAMSHIDHAR REDDY BODA

B. S., Osmania University, 2008

Kansas City, Missouri

2011

APPROVAL PAGE

The faculty listed below, appointed by the dean of the school of computing and engineering, have examined a thesis titled "Network Partition For Switched Industrial Ethernet Using Combined Search Heuristics", presented by Vamshidhar Reddy Boda, candidate for the masters of science degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Xiaojun Shen, PhD, Committee Chair
Department of Computer Science Electrical Engineering

Vijay Kumar, PhD
Department of Computer Science Electrical Engineering

Praveen R. Rao, PhD
Department of Computer Science Electrical Engineering

NETWORK PARTITION FOR SWITCHED INDUSTRIAL ETHERNET USING COMBINED SEARCH HEURISTICS

Vamshidhar Reddy Boda, Candidate for the Master of Science Degree
University of Missouri-Kansas City, 2011

ABSTRACT

A large industrial company needs a cost efficient telecommunication network to support heavy telecommunication needs among its different departments which are distributed in various locations. Because of the huge amount of daily communications, the network designer must partition the communicating devices into subnets each of which is supported by a high speed Ethernet. Then, the subnets are connected by a second level switch device called controller which handles inter-subnet communications. An optimization problem is how to partition *n* communicating devices into *k* groups such that the amount of intra-network traffic is balanced among the *k* groups and at the same time the inter-network traffic is minimized for a given traffic demand. This problem is known as the Network Partition Problem (NPP).

The NPP problem has been studied by some researchers, but because of its NP-hardness, only limited progress has been reported by two recent papers. The later one slightly improved on the results obtained by the previous one, and both papers used genetic algorithms.

This thesis investigated the NPP problem and concluded by extensive tests that it is very difficult to improve further if we purely follow the method of genetic algorithms. Motivated by searching for new approaches, this thesis tried another evolutionary algorithm,

i.e., the simulated annealing (SA) to see any hope to get a breakthrough. Encouraging results were obtained for some cases but not show overall superiority. Finally, this thesis investigated the approach that combines these two methods in searching for a better result. Extensive simulations demonstrated that this method work efficiently. By the combination of these two methods, we obtained obvious improvements on previous published results. This approach studied in this thesis can be applicable to practically solving other NP-hard problems also.

TABLE OF CONTENTS

LIST OF EQUATIONS

Equation

LIST OF IILUSTRATIONS

Figure

LIST OF TABLES

Table

CHAPTER 1

INTRODUCTION

A large industrial company needs a cost efficient telecommunication network to support heavy telecommunication needs among its different departments which are distributed in various locations. Because of the huge amount of daily communications, the network designer must partition the communicating devices into subnets each of which is supported by a high speed Ethernet. Then, the subnets are connected by a second level switch device called controller which handles inter-subnet communications. An optimization problem is how to partition *n* communicating devices into *k* groups such that the amount of intra-network traffic is balanced among the *k* groups and at the same time the inter-network traffic is minimized for a given traffic demand. This problem is known as the Network Partition Problem (NPP).

Evolutionary algorithms are stochastic search optimization techniques which simulate the natural biological evolution process through computer technology. They are applied extensively in variety of problem domains, which has a large search space. One such problem is NPP. These techniques, through a repeated iteration of search for a better solution, can obtain the optimal or near-optimal solution (Zalzala and Fleming). Of the several search methods, we have chosen Genetic Algorithm (GA) and Simulated Annealing (SA) to target the problem, as they proved to be superior to many other heuristics.

NPP is a multi-objective optimization, NP-Complete problem (Zhang and Zhang). The main objective of network partition is to partition the network into sub-networks, based on a communication relationship between communication nodes, such that aggregate inter-

communication is minimized, and simultaneously make the traffic in the network evenly distributed across the sub-networks. The research community has proposed a number of GAs to optimize the network design for a general communication network (Hsinghua, G., & Chao-Hsien, 2001; C., G., & H., 2000; R. & M., 1996; C.A., D. H., & M., 1994; M., R., & S.S., 2001; S. & G., 1998; H., M., T.L., & E., 1997; H. & K., 2001). Zhang & Zhang, 2007 did more a careful work on network partition especially in switched-industrial Ethernet by considering real-time characteristics.

Q. Zhang and W. Zhang proposed a mathematical model for network partition and applied GA method, based on their proposed mathematical model, to optimize the network partition for switched industrial Ethernet. Their work considered more real-time characteristics like the existence of controller and one-way communication characteristic of field devices. Apart from these characteristics, they have also considered a more general communication relationship between field devices which reflect the practical instances more accurately. But, their work has wrongly formulated the equation for total traffic burden calculation.

This work has corrected mathematical model formulation of a previous work and implemented GA with the corrected formulation. Solving NPP through GAs has been gradually accepted by specialized literature. Circumstantially, many methods and techniques based on GA have been proposed. This work focused to solve the problem using the combination of two search heuristics: Genetic Algorithm and Simulated Annealing. The result of a combination approach seems promising.

CHAPTER 2

MODEL DESCRIPTION AND PROBLEM DEFINATION

2.1 Model Description

This section presents the model description of network partition. Network partition mathematical model was proposed by Q. Zhang and W. Zhang, 2007.

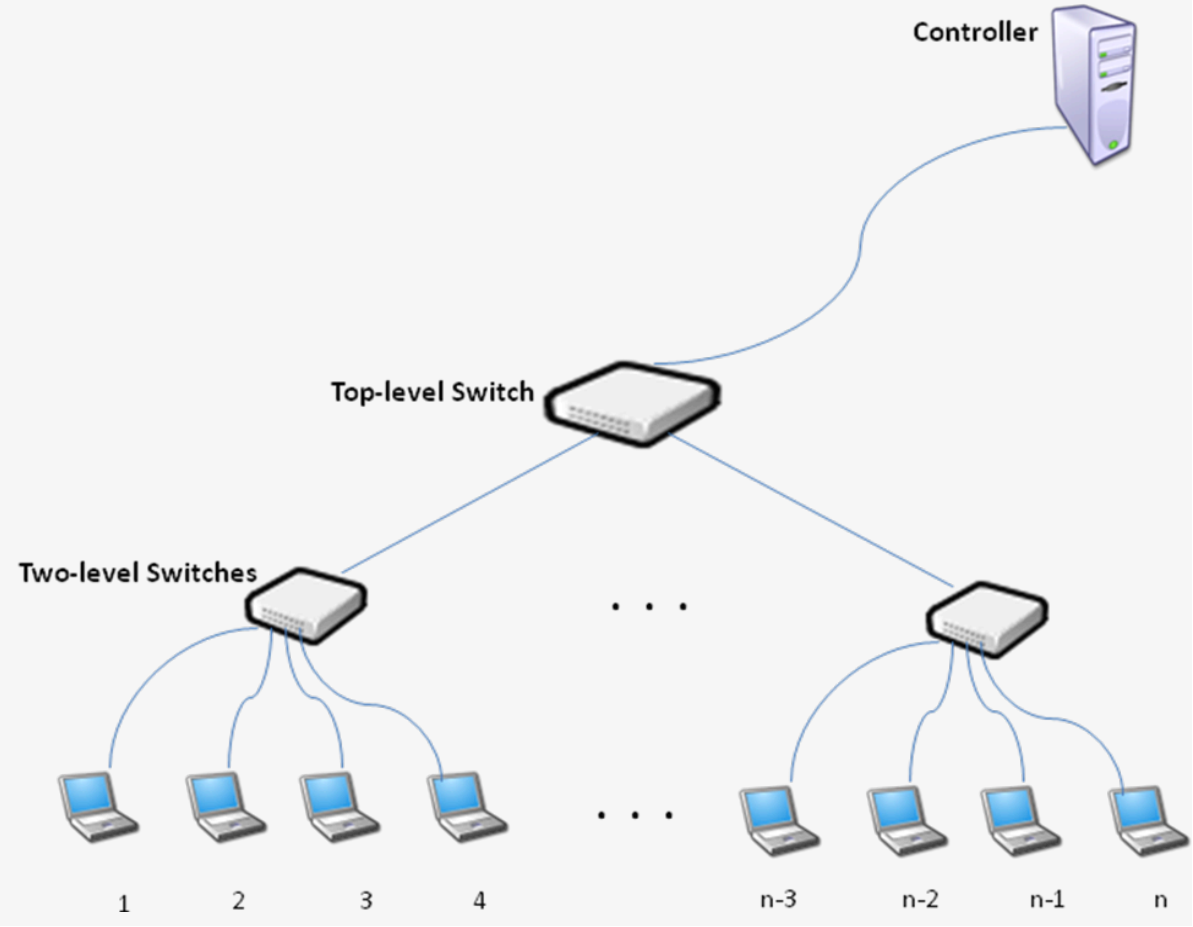


Figure 1 – The Switched Industrial Ethernet With 2-Level Tree Topology.

Figure 1 depicts the switched industrial Ethernet model with two-level tree topology. From the figure, Controller exchanges information with all devices on the factory floor. Since, it occupies a large ratio of communication; it is connected independently to a top-level switch. Since, number of switch ports is a constraint; devices must be partitioned into sub-networks, by connecting them to two-level switches as shown in the figure. These two-level switches are further connected to top-level switch. The network partition in communication network generally follows two principles (Songerwala):

1. Divide those devices having closer communication relationships into the same sub-network, such that intra-network communication is maximized and inter-network communication is minimized. This principle reduces the traffic forwarded by the top-level switch, therefore, improves the average transfer delay.
2. Balance the communication over the resultant sub- networks, such that the total traffic burden is evenly distributed in respective sub-networks. This principle prevents the network performance from being significantly influenced if new devices are added to an arbitrary sub-network.

Network partition is based on communication relationship between the nodes. This relationship is specified in the Traffic Matrix. The communication relationship or the data for traffic matrix may be estimated from the previous data. Generating traffic matrix is out of this work. This work assumes that traffic matrix is already given.

Let, $i=\{1,2,3....,n\}$ represent the field devices and $n+1$ represent the controller, then traffic matrix can be constructed as following:

$$\begin{bmatrix} 0 & a_{12} & \dots & a_{1n} & a_{1,n+1} \\ a_{21} & 0 & \dots & a_{2n} & a_{2,n+1} \\ \dots & \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & 0 & a_{n,n+1} \\ a_{n+1,1} & a_{n+1,2} & \dots & a_{n+1,n} & 0 \end{bmatrix} \quad (1)$$

Equation 1 - Traffic Matrix

a_{ij} is the communication measure from node *i* to node *j*. Assuming no devices send data to themselves, all the diagonal elements in the traffic matrix are zeros. Note that $(n+1)^{th}$ device is the controller. Controller contributes a lot to aggregate communication, so it is necessary to include in the traffic matrix.

2.2 Problem Definition

Let, $k = \{1, 2, 3 \dots K\}$ as sub-network, and define

$$x_{ik} = \begin{cases} 1. \ if \ node \ i \in subnet \ k, \\ 0, \ \ if \ node \ i \notin subnet \ k. \end{cases} \quad (2)$$

Equation 2 - Device to Sub-net *Relation*

If node *i* and node *j* belongs to the same subnet *k*, $x_{ik}(1 - x_{jk}) = 0$, otherwise $x_{ik}(1 - x_{jk}) = 1$. So, the aggregate inter-network communication among all sub-networks is given by the following equation, where *C* denotes the aggregate inter-network communication.

$$C = \sum_{k=1}^{K} \sum_{j=1, j \neq i}^{n} \sum_{i=1}^{n} x_{ik}(1 - x_{jk}) \quad (3)$$

Equation 3 - Aggregate Inter-network Communication Equation

Note: Controller is unconsidered while calculating aggregate inter-network communication, as the communication between the controller and all the sub-networks cannot be reduced.

For a given sub-network, traffic burden is data forwarded by the corresponding two-level switch. So, traffic burden for a sub-network *k*, is given by the following corrected equation:

$$w(k) = \sum_{j=1}^{n+1} \sum_{i \in k} a_{ij} + \sum_{i=1}^{n+1} \sum_{j \in k} a_{ij} - 2 * \sum_{j \in k} \sum_{i \in k} a_{ij} \quad (4)$$

Equation 4 - Total Traffic Burden Equation

In the above equation, *w(k)* is the total traffic burden of sub-network *k.* The former two terms of the equation are the sending and receiving data of sub-network *k*, respectively. Inner-subnet transmission has been calculated twice in former two terms. According to the definition of total traffic burden, it should be excluded. So, it is excluded by the last term of the Equation 4. One of the objectives is to make the network traffic be evenly distributed over the respective sub-networks. For this, we need to ensure that communication is evenly distributed. The communication imbalance among sub-networks is given by the following equation:

$$I = \max_{1 \le i \le K-1, i \le j \le K} |w(i) - w(j)| \quad (5)$$

Equation 5 - Communication Imbalance Equation

Where, *I* denoted maximum communication imbalance among all the sub-networks. The two design objectives can be combined using the Equations (3) & (5) into a single objective function as following:

$$f = C + \beta * I \quad (6)$$

Equation 6 - Optimization Function

In the above equation, *ß* is the weighted coefficient, which can be adjusted to compromise between two minimization objectives. Smaller *ß* attach relatively more emphasis on the minimization of inter-network communication, while larger *ß* attach relatively more

emphasis on the minimization of communication imbalance. The selection of the weighted coefficient, *ß,* directly affects the efficiency and solution of the optimization algorithm.

2.3 Errors corrected from the previous work

In the paper (Zhang and Zhang), traffic burden of the sub network is the data forwarded by the two-level switch of the sub-network, which means it must strictly exclude the elements that involve intra-network communication. But, Equation 4 of Zhang's paper does not satisfy the above statement. The Equation 4 in the section 2.2 is the correct equation that satisfies the total traffic definition.

Another error we have corrected is Communication imbalance equation. According to the definition, communication imbalance is the maximum difference of total traffic burden between any two sub-networks. The Equation 5 in the section 2.2 is the correct equation that satisfies the communication imbalance definition.

Table of Notations

Variable	Description
a_{ij}	Communication measure between node i to node j.
x_{ik}	Boolean variable; If node i belongs to sub-net k, x_{ik}=1, else x_{ik}=0.
C	Aggregate inter-network communication.
W	Total traffic burden.
I	Communication imbalance.
K	Number of Sub-networks.
n	Number of nodes in the network.
$n+1^{th}$ node	Controller
f	Fitness value
β	Weighted Coefficient

CHAPTER 3

OUR APPROACH

We have solved Network partition problem by the combination of two searching heuristics: Genetic Algorithm and Simulated Annealing. Their implementations are described in the following sections.

3.1 Genetic Algorithm

Genetic Algorithm (GA) mimics the evolution process of "survival of the fittest" in the nature. GA can be specified as follows:

$$GA = (N_{pop}, N_{gen}, N_d, N_s, P_c, P_m, f_{eval}, f_{sel})$$

where, N_{pop} is the number of individuals in the population, N_{gen} is the number of generations, N_d is the number of field devices, N_s is the number of sub-networks, P_c is the crossover probability, P_m is the mutation probability, f_{eval} is the fitness function, and f_{sel} is the selection rule. The Solution encoding and decoding is presented in the next section. Same encoding is used for both GA and SA.

3.1.1 Implementation Description

The implementation of GA Method is summarized in the Table 1.

Table 1- Algorithm Describes GA Methods Used In Our Approach

Algorithm 1 GA Method

```
1:  procedure GAMETHOD(N_pop, N_gen, N_d, N_s, P_c, P_m, f_eval, f_sel)
2:      Population : pop_g ← INITIALIZEPOPULATION(N_d, N_s)
3:      REPEAT N_gen times {
4:          CALCULATEFITNESS(pop_g, f_eval)
5:          Chromosome : weakest ← FINDWEAKEST(pop_g)
6:          Chromosome : fittest ← FINDFITTEST(pop_g)
7:          Population : pop_i ← SELECTIONOPERATION(pop_g, f_sel)
8:          Population : pop_c ← CROSSOVEROPERATION(pop_i, P_c)
9:          Population : pop_m ← MUTATIONOPERATION(pop_c, P_m)
10:         Population : pop_n ← ELITEOPERATION(pop_m, weak, fittest)
11:         pop_g ← pop_n
12:     }
13:     Chromosome : solution «- FINDFITTEST(pop_g)
14:     return solution
15: end procedure
```

In the above pseudo code, *Chromosome* is the solution representation as described in 3.1.3; *Population* is the array of chromosomes. The variables *weakest* and *fittest* represent weakest and strongest individuals in the current generation, respectively. Let pop_g, pop_i, pop_c, pop_m and pop_n represent current population, intermediate population, crossover applied population, mutation applied population, and next generation population.

The function INITIALIZEPOPULATION() returns the population with random, but valid individuals. For a valid individual, refer section 3.1.3. The function CALCULATEFITNESS() calculates the fitness value for each individual in the given population, based on the fitness function described in 3.1.2. The functions FINDWEAKEST() and FINDFITTEST() returns the weakest chromosome and fittest chromosome in the given population. The function SELECTIONOPERATION() returns the intermediate or selected or reproduced population based on the selection rule. In this approach, the selection rule is Stochastic Remainder Sampling. The function CROSSOVEROPERATION() returns the population that is formed after applying crossover genetic operation with a given probability. In this approach, we have selected uniform crossover. The function MUTATIONOPERATION() returns the population after applying mutation genetic operator with a given probability. The function ELITEOPERATION() returns the population after applying elite genetic operator.

3.1.2 Fitness Function

Fitness function helps in evaluating the fitness of an individual or chromosome. This fitness function is primarily used by selection operator. Highly fit individuals have the higher probability of getting selected for mating, whereas less fit individuals have lesser probability. The problem we are working on is global minimization problem. To facilitate selection operation, this problem is converted into maximization problem, by transforming Equation 6 into Equation 7

$$F = \begin{cases} U - \sum_{k=1}^{K} \sum_{j=1, j\neq i}^{n} \sum_{i=1}^{n} x_{ik}(1 - x_{jk}) a_{ij} \; - \beta \max_{0 \le x \le 1} |w(k+1) - \; w(k)|, & f < U, \\ 0, & f \geq U. \end{cases}$$

Equation 7 - Fitness Function

where *F* stands for fitness value, *U* is an appropriate positive number to ensure the fitness of all good individuals is positive in feasible solution space. *U* is used to adjust selection pressure of GA. Selection pressure is inversely proportional to *U*. If the selection pressure is less, then evolution process is prevented from premature convergence thereby avoiding to get trapped in local minima. But, if *U* is too large, i.e. if selection pressure is too less, then the process of evolution is very slow.

3.1.3 Solution Representation

According to the mathematical model presented in the Chapter 2, the solution encoding should allow us to decode the following information:

1. Aggregate Inter-network communication.
2. Total traffic burden for each sub-network.
3. Communication Imbalance among all the sub-networks.

The encoding can be categorized into two types: direct and indirect encoding. Direct encoding is verbose and can be read directly, while with indirect encoding, a decoding algorithm is used to decode the required information. Since, direct encoding utilizes lot of computation resources; we have used indirect encoding in our approach.

Indirect encoding is an integer encoding, which was used by many scholars (Krommenacker, Divoux, & Rondeau, 2002; Pierre & Houéto, 2002; Quintero & Pierre, 2003; Salcedo-Sanz & Yao, 2004). The solution of NPP is encoded into an array of integers.

The array size is equal to number of field devices used in the network. Each cell of the array corresponds to a field device. The i^{th} element of the array is *k,* when field device *i* is assigned to sub-network *k.* For example, the field devices $n = \{1, 2\ 3, 4, 5, 6, 7, 8, 9\}$ are assigned to three sub-networks $k = \{1, 2, 3\}$, and one possible individual could be [1 2 2 3 1 2 2 3 1]. Then the corresponding network partition is {1, 5, 9}, {2, 3, 6, 7} and {4, 8}.

According to GA specialized literature, each solution is called as a chromosome or an individual. Given an individual, it can be either valid or invalid. This work assumed an individual to be valid, if it satisfies the following constraints:

1. Number of devices assigned to each sub-network is less than number of ports of a switch.
2. Number of devices assigned to all sub-networks is roughly equal.

3.1.4 Solution Decode Method

The Solution decode method allows to calculate the fitness of a chromosome by decoding the individual. According to the mathematical model presented in the Chapter 2,

$$f = C + \beta * I$$

where, *f* is the objective value, *C* is aggregate inter-network communication, *I* is communication imbalance and *ß* is weighted coefficient. The procedure CALCULATEFITNESSVALUE() returns the fitness value of an individual. This procedure is presented in Table 2.

Table 2 - Algorithm describes Fitness value calculation for a given individual

Algorithm 2 Calculate Fitness Value of an Individual

```
1:  procedure CALCULATEFITNESSVALUE (chromosome)
2:      Real: C ← COMPUTENETWORKCOMMUNICATION(chromosome)
3:      Real: I ← COMPUTECOMMUNICATIONIMBALANCE(chromosome)
4:      Integer: F ← C + ß* I
5:      return F
6:  end procedure
```

In the above pseudo code, the function COMPUTENETWORK-COMMUNICATION() returns the aggregate inter-network communication for a given chromosome and the function COMPUTECOMMUNICATIONIMBALANCE() returns the communication imbalance from a given chromosome. The procedures COMPUTENETWORKCOMMUNICATION() and COMPUTECOMMUNICATIONIMBALANCE() are presented in Table 3 and Table 5, respectively.

Table 3 - Algorithm Describes Aggregate Inter-network Communication Calculation For A Given Individual

Algorithm 3 Compute aggregate Inter-network communication

```
1:  procedure COMPUTENETWORKCOMMUNICATION (chromosome)
2:      Integer: C ← 0
3:      Matrix: deviceMatrix[Nd][Ns] ← BUILDDEVICEMATRIX(chromosome)
4:      FOR k ← 1 TO Ns STEP 1
5:          FOR j ← 1 TO Ns  STEP 1
6:              FOR i ← 1 TO Ns STEP 1
7:                  IF (i NOTEQUAL j )
8:                      Integer: x ← 0
9:                      x ← (deviceMatrix[i][k] * (1 - deviceMatrix[j][k])) * trafficMatrix[i][j]
10:                     C ← C + x
11                  ENDIF
12:             NEXT i
13:         NEXT j
14:     NEXT k
15:     return C
16:  end procedure
```

In the above pseudo code, the function BUILDDEVICEMATRIX() builds a Boolean matrix called Device Matrix. It is an N_d-*by*-N_s matrix. The $(i, k)^{th}$ entry of the matrix is 1, if the field device *i* belongs to sub-network *k*. For example, if the entry (2, 3) of the matrix is 1, then field device 2 belong to sub-network 3. Traffic matrix gives the communication relationship between two communication nodes in the network. Refer Chapter 2 for traffic matrix details.

Table 4 - Algorithm Describes Building Device Matrix For A Given Individual

Algorithm 4 Build Device Matrix

```
1:  procedure BUILDDEVICEMATRIX (chromosome)
3:      Matrix: deviceMatrix[Nd][Ns] «- 0
4:      FOR i ← 1 TO Nd STEP 1
5:          deviceMatrix[i][chromosome[i]] ← 1
6:      NEXT i
7:      return deviceMatrix
8:  end procedure
```

Table 5 - Algorithm Describes Calculating Communication Imbalance among Sub-networks

Algorithm 5 Compute Communication Imbalance

```
1:  procedure COMPUTECOMMUNICATIONIMBALANCE (chromosome)
2:      Array: W[Ns] ← 0
3:      Integer: I ← 0
4:      FOR k ← 1 TO Ns STEP 1
5:          W[k] ← CALCULATETRAFFICBURDEN(chromosome, k)
6:      NEXT k
7:      FOR i ← 1 TO Ns - 1 STEP 1
8:          FOR j ← i TO Ns - 1 STEP 1
9:              I ← MAX {ABS(W(i) - W(j))}
10:         NEXT j
11:     NEXT i
12:     return I
13: end procedure
```

The pseudo code described in Table 5, the variable W is an array of total traffic burdens. The function CALCULATETRAFFICBURDEN() will return the total traffic burden of sub-network k. The k^{th} cell of array W holds the total traffic burden of sub-network k. The procedure CALCULATETRAFFICBURDEN() is presented in Table 6.

Table 6 - Algorithm Describes Calculating Total Traffic Burden Of A Sub-network

Algorithm 6 Compute Communication Imbalance

```
1:  procedure CALCULATETRAFFICBURDEN (chromosome, k)
2:      Integer: value ← 0
3:      FOR i ← 1 TO SIZE(chromosome) STEP 1
4:          IF(chromosome[i] EQUALS k)    'if field device i belongs to sub-network k'
5:              FOR j ← 1 TO SIZE(chromosome) STEP 1
6:                  value ← value + trafficMatrix[i][j]
7:              NEXT j
8:          ENDIF
9:      NEXT i
10:     FOR i ← 1 TO SIZE(chromosome) STEP 1
11:         FOR j ← 1 TO SIZE(chromosome) STEP 1
12:             IF(chromosome[j] EQUALS k)    'if field device j belongs to sub-network k'
13:                 value ← value + trafficMatrix[i][j]
14:             ENDIF
15:         NEXT j
16:     NEXT i
17:     FOR i ← 1 TO SIZE(chromosome) STEP 1
18:         IF(chromosome[i] EQUALS k)    'if field device i belongs to sub-network k'
19:             FOR j ← 1 TO SIZE(chromosome) STEP 1
20:             IF(chromosome[j] EQUALS k)    'if field device j belongs to sub-network k'
21:                 value ← value – 2 *  trafficMatrix[i][j]
22:             ENDIF
23:             NEXT j
24:         ENDIF
25:     NEXT i
26:     return value
27: end procedure
```

3.1.5 Population Initialization

Population initialization creates the first generation of population to start the evolution process. The function INITIALIZEPOPULATION() creates the first generation population with random, but valid individuals. Twice the number of N_{pop} of valid chromosomes is generated. Then, the first half of better individuals is selected into the initial population.

3.1.6 Genetic Operators

There are four genetic operators applied for each generation of population. They are listed as follows:

1. Selection Operator
2. Crossover Operator
3. Mutation Operator
4. Elite Operator

Selection operator is used to select the stronger individuals and remove the weaker ones from the population for reproduction. The more fit they are, the more chances they can be selected for reproduction. There are several types of selection methods such as tournament selection, roulette wheel selection, deterministic sampling and stochastic remainder sampling (Andrade, Luciano and Carlos Henrique Nogueira). In our approach, we have used stochastic remainder sampling with roulette wheel selection. In the stochastic remainder sampling, the average fitness, f_{avg}, of the population is calculated. Then, fitness of the each individual is divided by f_{avg}. If this value is greater than 1, then individual is selected into the intermediate population, which is used for reproduction. The empty slots of the population are filled by roulette wheel selection method.

Crossover genetic operator is used to reproduce new chromosomes from the intermediate population. New chromosomes are reproduced by exchanging the partial chromosome of mated parents in a random mode. The motive of crossover operator is to maintain the quality of solution set, while probing new area in the feasible solution space. There are several flavors of crossover operator. In our approach, we have used uniform crossover. In the uniform crossover, a random array of bits, whose size is same as chromosome, called mask vector is considered. In order to produce a child, one must simply choose the mask vector, bit by bit. If the bit chosen is 0, then parent A is picked up else parent B. For every crossover operation two child chromosomes are generated. During the generation of the first child, the value 0 in the mask indicates to consider the bit in the corresponding position from the parent A and the value one indicates to consider the bit in the corresponding position from the parent B. During the generation of second child, it is vice versa, which means that mask bit as 0 indicates that bit in the corresponding position from parent B is copied and 1 indicates that bit in the corresponding position from parent A is copied. Figure 2 is an example of uniform crossover for a scenario of 4 sub-networks and 12 field devices.

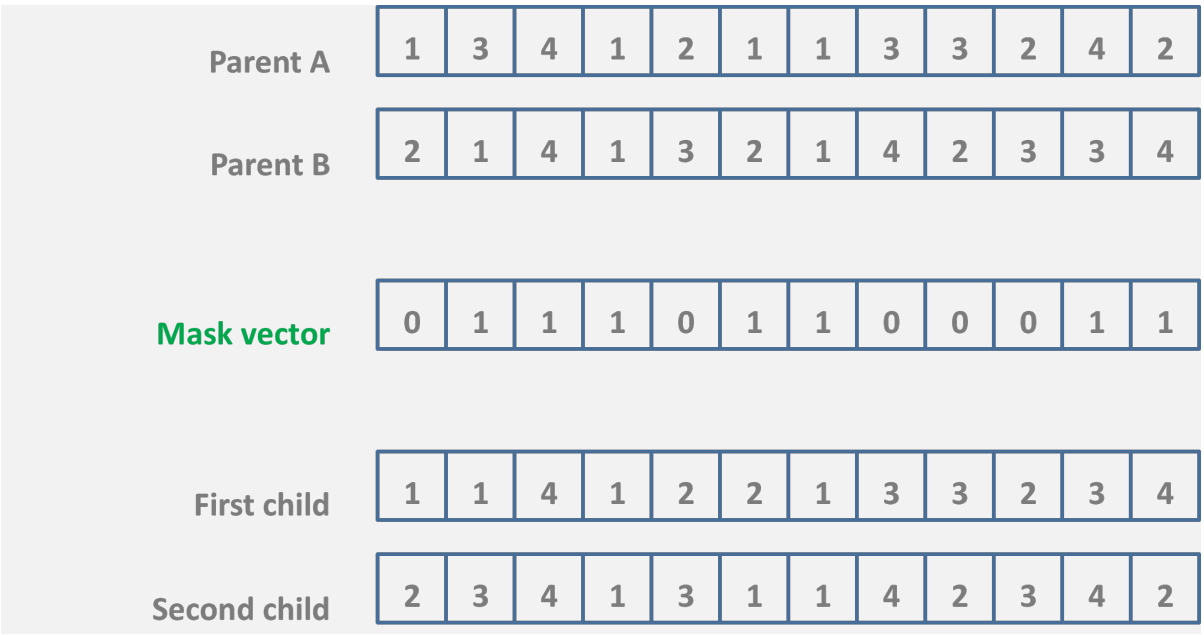


Figure 2 - Uniform Crossover Operator.

After each crossover operation, the offspring are immediately checked whether they are valid. If they are invalid, they are discarded and operation is re-applied. If valid offspring cannot be obtained in 20 reties, crossover operation is terminated. Crossover operation between two chromosomes is performed based on the probability called crossover probability, which is specified by P_c.

Another genetic operator, we have used in this approach, is mutation operator. Based on the mutation probability, P_m, the allele of a gene is replaced by another value in a stochastic mode. Operation is retried, if chromosome generated is invalid.

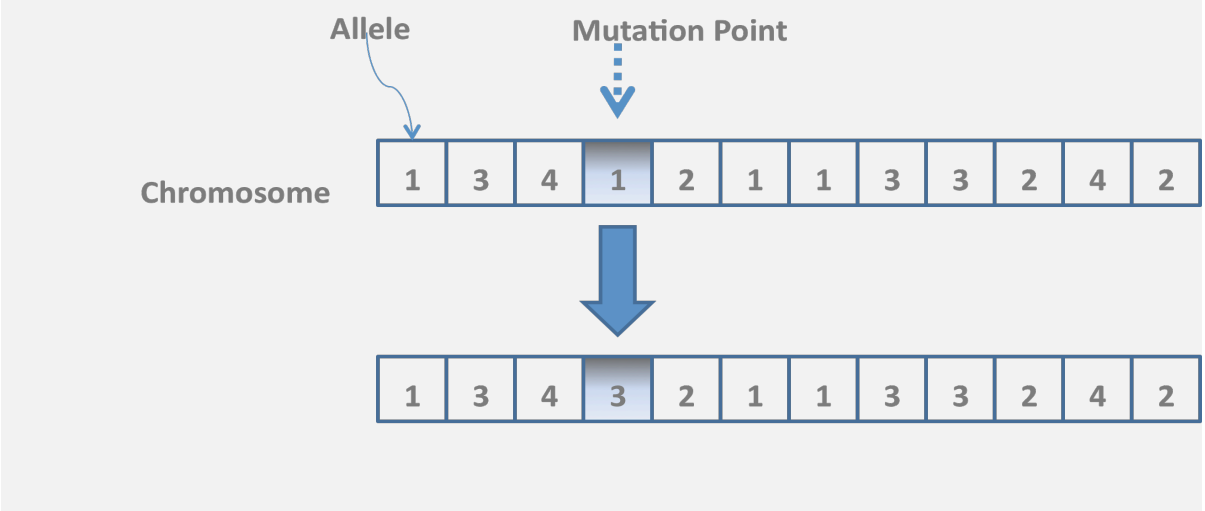


Figure 3 - Mutation Operator.

A good solution may be found in the early phases of the evolution process, but it may be deleted as evolution progresses. Elitism Technique is used to remember the best solution. In generation of population, the weakest individual is replaced by fittest individual.

3.2 Simulated Annealing

Another search technique used in this approach is Simulated Annealing (SA). SA belongs to a class of algorithms called probabilistic hill-climbing (Adler) which alters the inferior solutions acceptance probability dynamically. Hill climbing always selects the better move, while SA selects a random move from the neighbourhood based on acceptance probability. If the move is better than its current position, then simulated annealing always takes it. If the move is worse, then it will be accepted based on dynamically changing probability called Acceptance probability.

SA can be specified as follows:

$$SA = (S_f, T_f, N_d, N_s, P_m, f_{eval})$$

where, S_f is annealing scheduling factor, T_f is the final temperature, N_d is the number of field devices, N_s is the number of sub-networks, P_m is the move probability, f_{eval} is the fitness function. Same encoding is used for both GA and SA. For details, refer section 3.1.3 and 3.1.4.

3.2.1 Implementation Description

The implementation of SA Method is summarized in the Table 7.

Table 7 - Algorithm Describes SA Methods Used In Our Approach

Algorithm 7 SA Method

```
1:  procedure SAMETHOD(Sf, Tini, Nd, Ns, Pm, feval)
2:      Chromosome: currChromosome ← INITIALIZECHROMOSOME (Nd, Ns)
3:      Real: Tcurr ← Tini
4:      REPEAT
5:        Chromosome : newChromosome ← MOVE(currChromosome )
6:         Double:firnessChange ← FITNESS(newChromosome) -
                                                  FITNESS(newChromosome)
7:        IF fitnessChange is POSITIVE
8:          currChromosome ← newChromosome
9:        ELSE IF (ACCEPT(fitnessChange, Tcurr) is TRUE )
10:         currChromosome ← newChromosome
11:       Tcurr ← Tcurr * Sf
12:     Until Tcurr ≠ ε
13:     return currChromosome
14:  end procedure
```

In the above pseudo code, "Chromosome" is the solution representation as described in the section, 3.1.3. The variable T_{curr} represents the current temperature. The function INITIALIZECHROMOSOME () returns the chromosome with a random, but valid individual. The function FITNESS() return the fitness value of the chromosome, based on the fitness function described in f_{eval}. The function MOVE() moves the chromosome to a new position. This operation is similar to Genetic Algorithm's mutation operator. The function ACCEPT() is called when the chromosome created by Move operator is inferior to the current chromosome. The function ACCEPT() returns the decision, to accept the new

chromosome, based on acceptance probability. Acceptance probability is changes dynamically. It is calculated by the following equation:

$$P(A) = e^{-(\frac{fitnessChange}{temperatue})}$$

3.3 Combination

Most of the research community's work targeted NPP using Genetic Algorithms. Our preliminary research also used Genetic Algorithm as function optimizer. But, the results obtained did not show much improvement. This was the basis for our motivation to approach the problem with a different search technique. We have noticed that GA evolution process suffers from the problem of getting stuck at local minima or premature convergence. So, we have tried to overcome problem of premature convergence by using SA.

SA solves the problem of premature convergence by allowing worse moves (lesser quality) to be taken at some points of the process, i.e., it allows some downhill steps, so that, it can escape from local minima. So, SA was applied to NPP.

SA Method outperformed GA Method at higher values of weighted coefficient, in the fitness function, *f*. But, on the other hand, SA Method poorly performed at lower values of weighted coefficient. Based on this observation, combination of the 2 methods: GA Method and SA Method should improve the results overall. So, we have used the combination of GA Method and SA Method to solve the problem.

For each value of weight coefficient, β, in the fitness function, GA Method and SA Method are invoked. The best individual among the results of two methods is selected as the Pareto solution for a given weighted coefficient. The Pareto solution obtained is not from complete interleaved method of Genetic Algorithm and Simulated Annealing. An attempt

was made to design truly interleaved methods of Genetic Algorithm and Simulated Annealing. Due to the following factors, it is difficult to design a truly interleaved method:

1. Both Genetic Algorithm and Simulated Annealing methods have different input parameters.
2. Genetic Algorithm maintains population of individuals for each generation, while Simulated Annealing does not maintain any population.
3. Incompatibly of runtime factors in switching the process between Genetic Algorithm and Simulated Annealing.

CHAPTER 4

RESULTS

Our approach solved NPP by the combination of two methods: Genetic Algorithm with uniform crossover, mutation, elite operators and stochastic remainder sampling as the selection rule, and Simulated Annealing. We compare our method with the method proposed by Q. Zhang & W. Zhang in the paper "Network partition for switched industrial Ethernet using genetic algorithm", 2007, based on the results.

We have implemented Q & W. Zhang's method for data collection. The Data collection experimental was based on their experimental design and parameters. 40 field devices were considered for network partition for the network topology as shown in the Figure 1. 4 switches each with 16 ports are considered as two-level switches. The traffic matrix (see the Appendix) was considered from Zhang's work. This matrix was randomly generated by modeling the communication characteristics of industrial Ethernet. The traffic load between arbitrary two nodes is represented using integers from 1 to 10, i.e. the matrix element $x_{ij} \in \{1, 2, 3, \ldots, 10\}$. They are used to simulate the network traffic in a real switched industrial Ethernet, where $x_{ij} = 1$ indicates node i sends a minimum Ethernet frame (64 bytes) to node j at a period of 0.01 s, and traffic loads represented by other integers are in direct proportion to their values.

GA parameters considered are set as follows: $pop_{size} = 40$, crossover probability, $P_c = 0.8$, mutation probability, $P_m = 0.01$, generations, $N_{gen} = 1000$. The GA routine is executed twenty times, each time a different weight coefficient $\beta(t) = 0.001 * 1.6^t$, $1 \leq t \leq 20$)

is set for the fitness function. Table 8 displays the data collection after implementing Zhang's methods with their mathematical model:

Table 8 – Data Collection From Zhang's GA Method With Their Mathematical Model

Weight Coefficient(β)	Aggregate inter-network communication(C)	Communication Imbalance(I)	Objective Value(f)
0.0016	1034	419	1034
0.0026	1071	364	1071
0.0041	995	487	996
0.0066	1070	276	1071
0.0104	1041	503	1046
0.0168	1050	346	1055
0.0268	995	257	1001
0.0429	1015	526	1037
0.0687	1037	71	1041
0.1099	1038	29	1041
0.1759	1084	180	1115
0.2814	1084	27	1091
0.4503	1141	10	1145
0.7205	1129	29	1149
1.1529	1154	5	1159
1.8446	1159	2	1162
2.9514	1152	10	1181
4.7223	1352	3	1366
7.5557	1250	2	1265
12.089	1357	6	1429

Above data is collected based on the mathematical model proposed by Zhang et al., 2007. This data was matching with results mentioned in their paper. But, their proposed mathematical model has an error in calculating total traffic burden. For details, view section 2.3. So, Zhang's GA method is re-implemented with rectified mathematical model. Table 9

displays the data collection after implementing Zhang's methods with rectified mathematical model:

Table 9 – Data Collection From Zhang's GA Method With Rectified Mathematical Model

Weight Coefficient(θ)	Aggregate inter-network communication(C)	Communication Imbalance(I)	Objective Value(f)
0.0016	1083	197	1083
0.0026	1032	194	1032
0.0041	1051	215	1051
0.0066	1038	270	1039
0.0104	1007	243	1009
0.0168	1030	177	1032
0.0268	992	246	998
0.0429	1029	150	1035
0.0687	1033	92	1039
0.1099	1005	183	1025
0.1759	1020	172	1050
0.2814	1048	60	1064
0.4503	1044	12	1049
0.7205	1038	20	1052
1.1529	1141	9	1151
1.8446	1116	11	1136
2.9514	1263	2	1268
4.7223	1239	5	1262
7.5557	1258	1	1265
12.089	1346	2	1370

The above data is graphically represented in the Figure 4

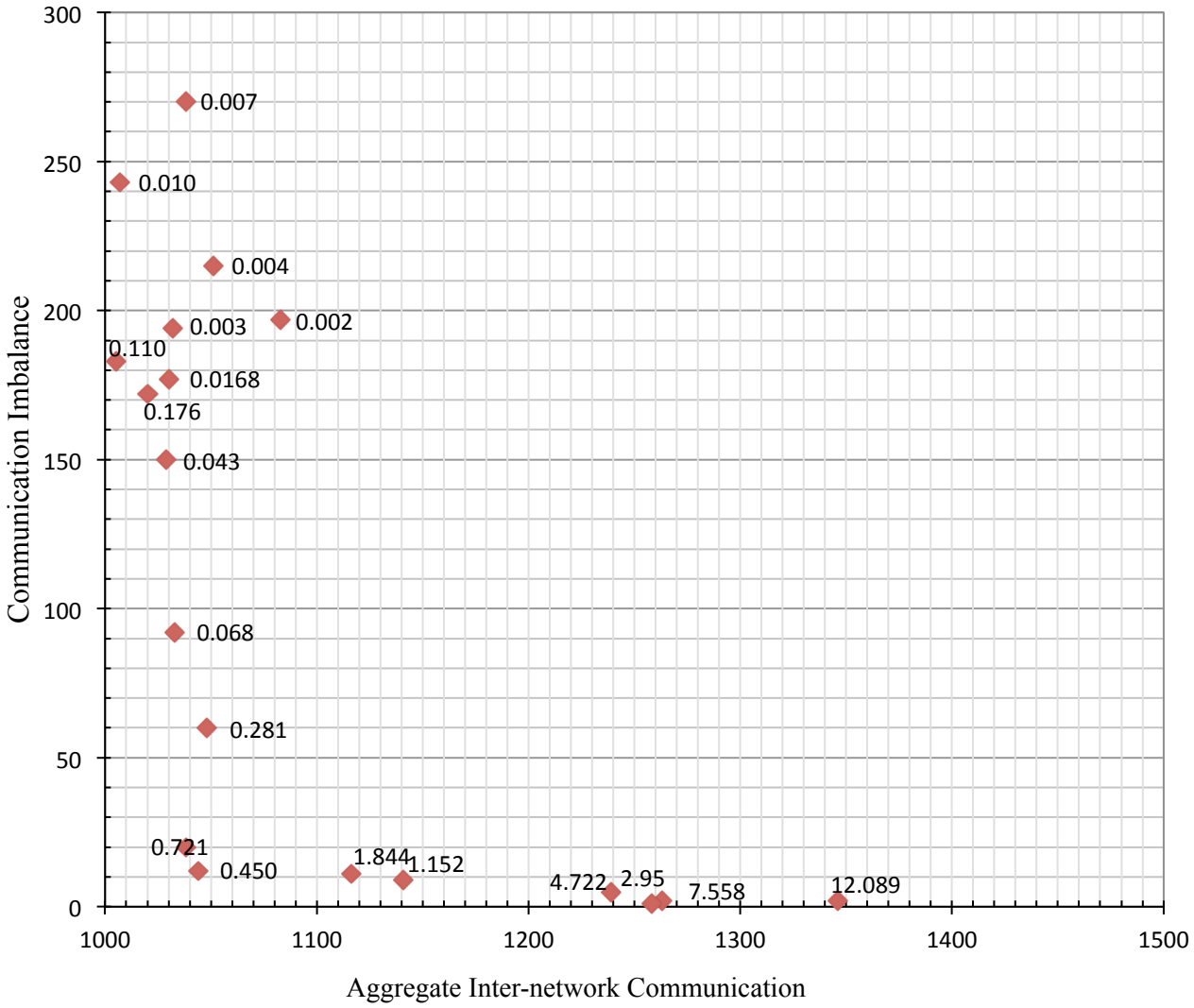


Figure 4 - The Obtained Pareto Solutions With Different Weight Coefficients, And The Pareto Solution With β=0.45 is Selected.

From the above graph, it can be observed that when β is increased, communication imbalance is reduced and aggregate inter-network communication is increased. But, this is not strictly followed because GA is a stochastic search technique, and it fails to ensure the optimal solution is found in all the cases. While selecting a Pareto solution, we need to

compromise upon communication imbalance and aggregate inter-network communication. Since, aggregate inter-network communication is our main optimization goal, the Pareto solution with $\beta = 0.4503$ is apparently a good selection.

GA evolution process details are depicted in the following graphs. Figure 7 shows that Objective values of best individuals decrease monotonically over the generations. Figure 5 and Figure 6 reflect that aggregate inter-network communication and communication imbalance, respectively, gradually decrease with evolution process, but also increase at some point over few generations, which indicates that two optimization goals compete with each other in the process of evolution.

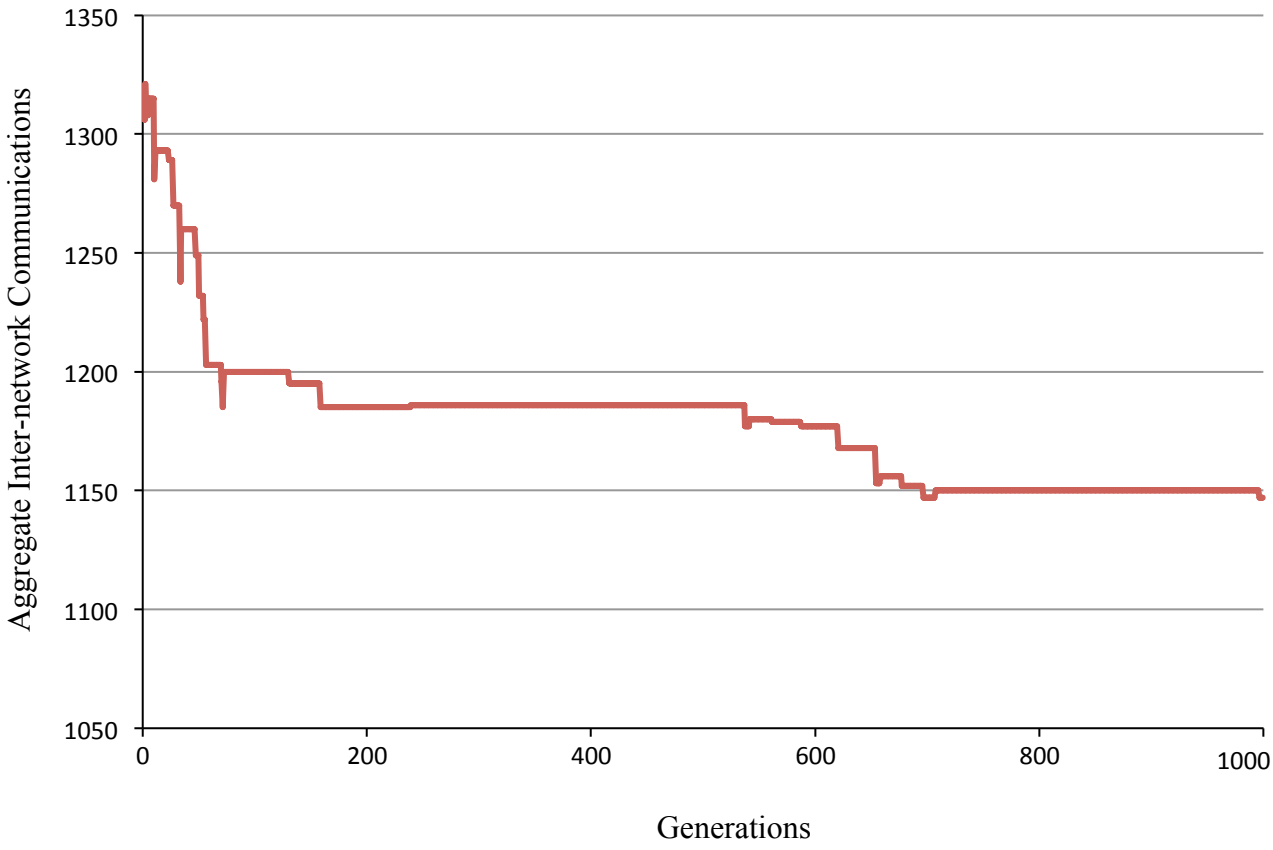


Figure 5 - Aggregate Inter-network Communication Of The Best Individual Over The Evolution Process

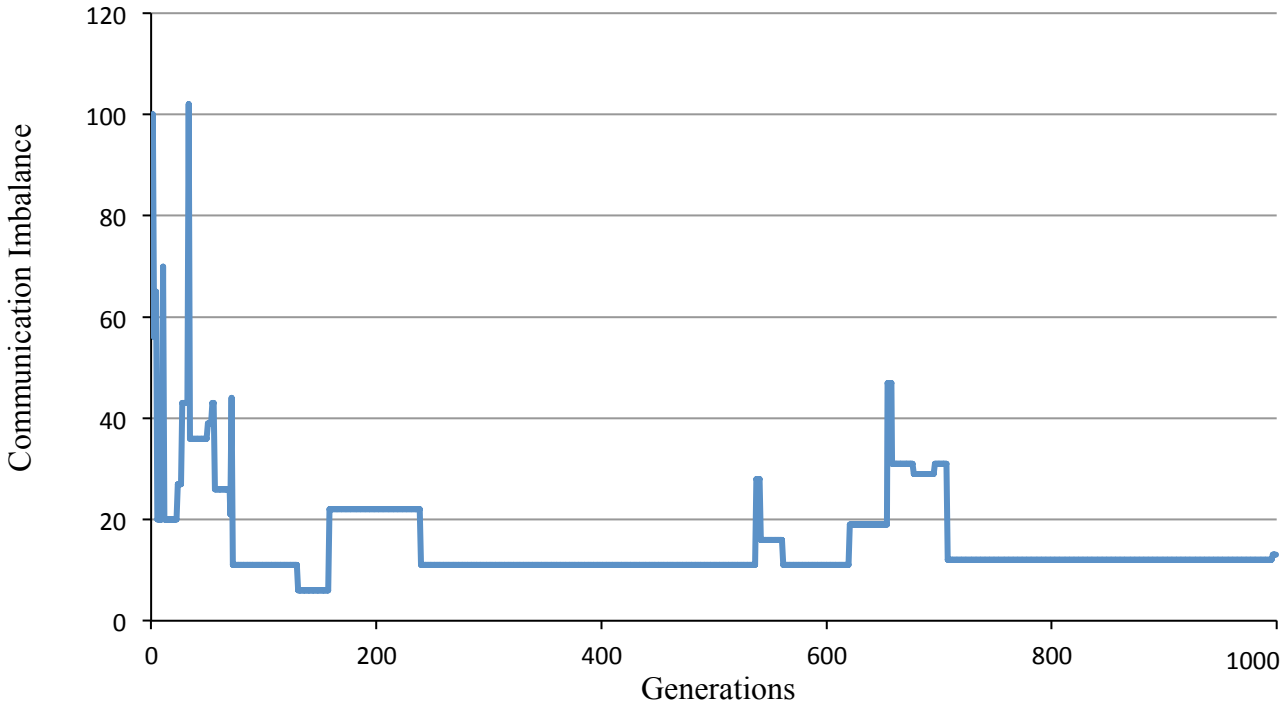


Figure 6 – Communication Imbalance Of The Best Individual Over The Evolution Process.

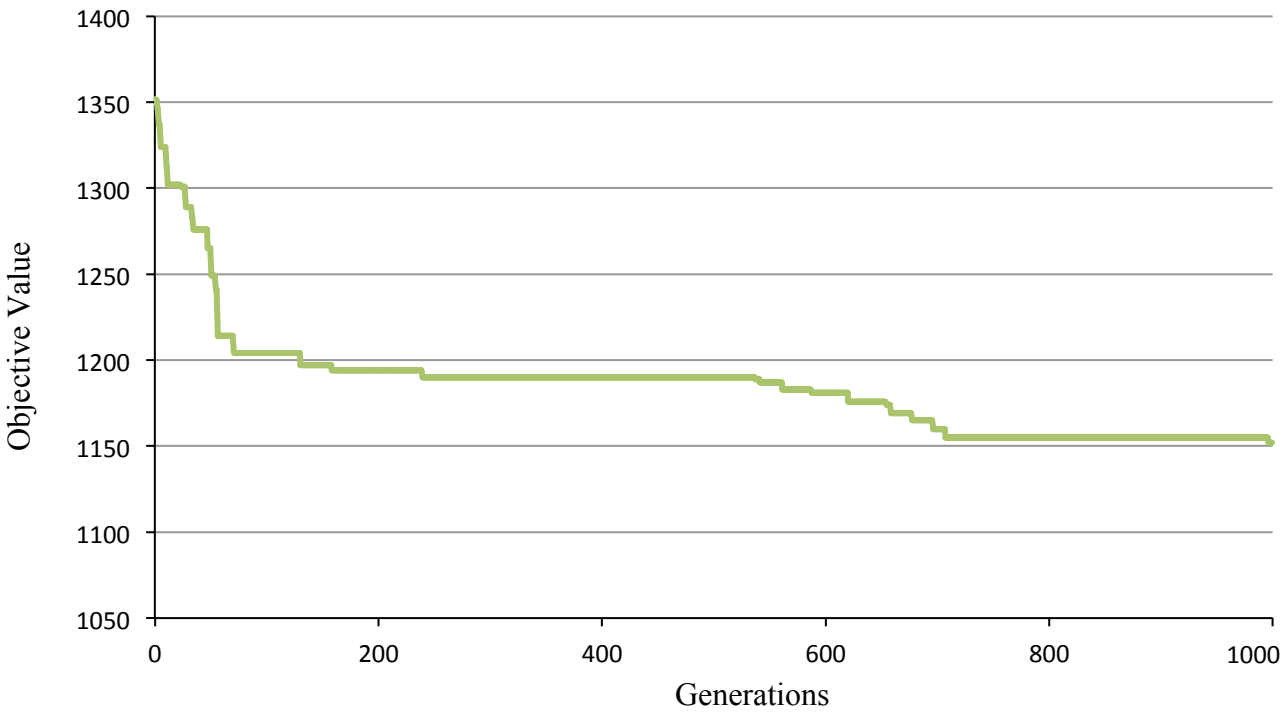


Figure 7 – Objective Value Of The Best Individual Over The Evolution Process.

From the data collection of Zhang's GA Method evolution process, we have noticed that their GA Method suffered from the problem of getting trapped into local minima or prematurely converging at the very early stages of evolution process. So, we further researched into their GA Method by decreasing the crossover probability, which was one of the remedies to avoid premature convergence. But, the results were degenerated. This motivated us to change the GA approach. Then, we have implemented our GA Method. Our GA Method is based on the following selection rule and genetic operators:

1. Uniform crossover operator.
2. Mutation operator.
3. Elite operator.
4. Stochastic remainder sampling.

For more details on our GA Method, see Chapter 3. For data collection, GA parameters considered are set as follows: $pop_{size} = 40$, crossover probability, $P_c = 0.8$, mutation probability, $P_m = 0.01$, generations, $N_{gen} = 1000$. The GA routine is executed twenty times, each time a different weight coefficient $\beta(t) = 0.001 * 1.6^t$, $1 \leq t \leq 20$) is set for the fitness function. Table 9 displays the data collection our GA Method.

Table 10 - Data Collection From Our GA Method

Weight Coefficient(β)	Aggregate inter-network communication(C)	Communication Imbalance(I)	Objective Value(f)	Improvement (%)
0.0016	1005	165	1005	7.202216066
0.0026	1018	185	1018	1.356589147
0.0041	998	250	999	4.947668887
0.0066	1013	224	1014	2.406159769
0.0104	1013	144	1014	-0.495540139
0.0168	1013	154	1015	1.647286822
0.0268	1023	101	1025	-2.705410822
0.0429	1011	134	1016	1.835748792
0.0687	989	248	1006	3.176130895
0.1099	995	202	1017	0.780487805
0.1759	1009	76	1022	2.666666667
0.2814	1031	37	1041	2.161654135
0.4503	1046	29	1059	-0.953288847
0.7205	1074	15	1084	-3.041825095
1.1529	1088	6	1094	4.952215465
1.8446	1147	9	1163	-2.376760563
2.9514	1264	3	1212	4.416403785
4.7223	1209	3	1223	3.090332805
7.5557	1247	2	1262	0.23715415
12.089	1327	1	1339	2.262773723

For each weight coefficient, we have compared best individual's objective values of two methods. Column 5 of Table 10 indicates the percentage of improvement compared with Zhang's GA Method. From Table 10, we can see that most of the Pareto solutions improved through our method. Figure 8 is the graphical representation of Table 10. From the graph, the Pareto solution with $\beta = 0.4503$ is apparently a good selection.

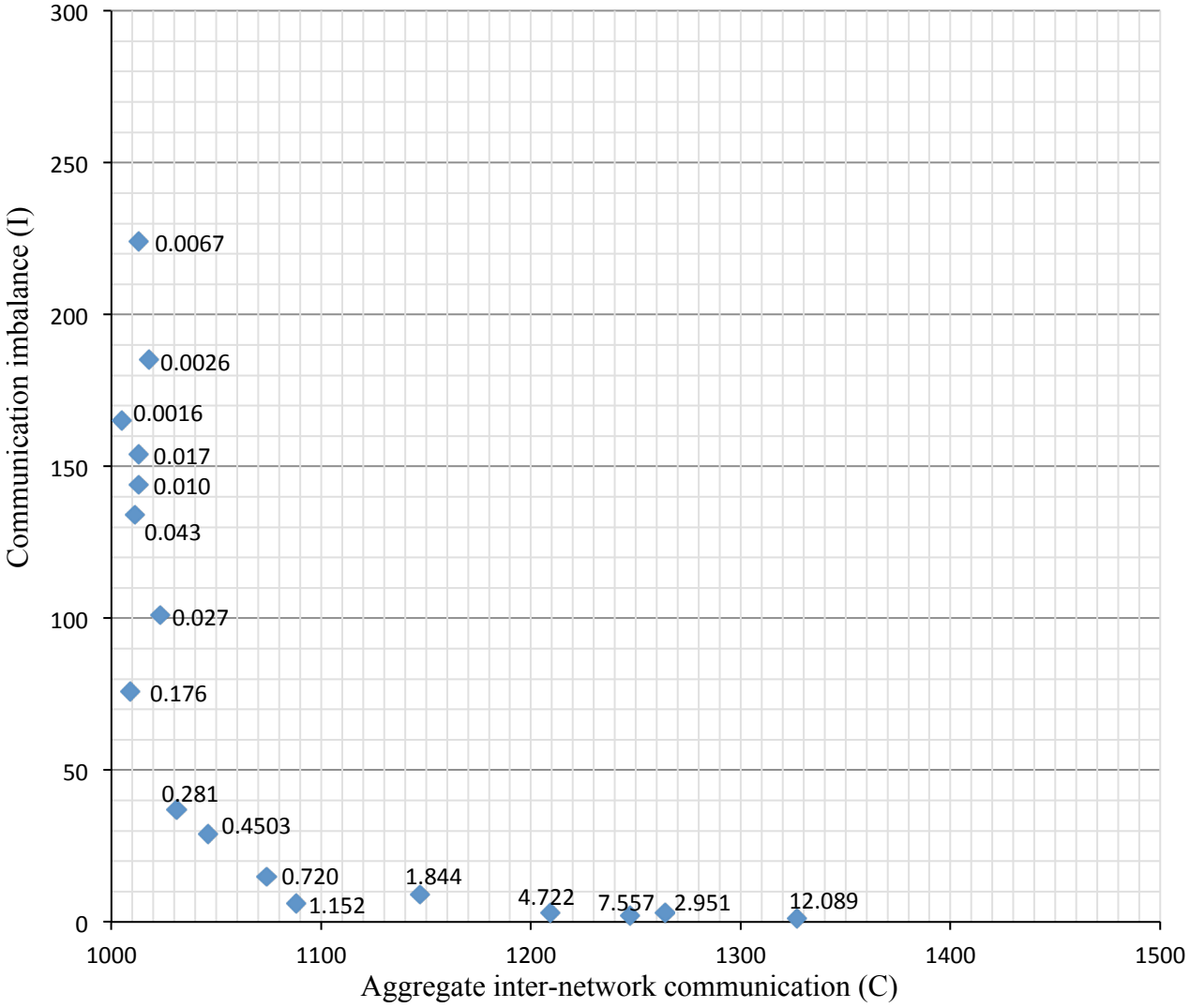


Figure 8 - The Obtained Pareto Solutions With Different Weight Coefficients, And The Pareto solution with β=0.4503 is selected.

Premature convergence is delayed through our method. Although, our GA Method could not remove the premature convergence, it has managed to avoid in the early phases of the evolution process. Figure 9 shows the comparison of Zhang's GA Method and Our GA Method's evolution process for the selected Pareto solution's weight coefficient, $\beta = 0.4503$ and above considered GA parameters. Zhang's GA Method used Two-point crossover and

our method used uniform crossover. From the Figure 9, we can see that Zhang's method stopped to find better solution from 311th generation, while our method is able find better solution even at 793rd generation. From this, we can say that premature convergence is delayed.

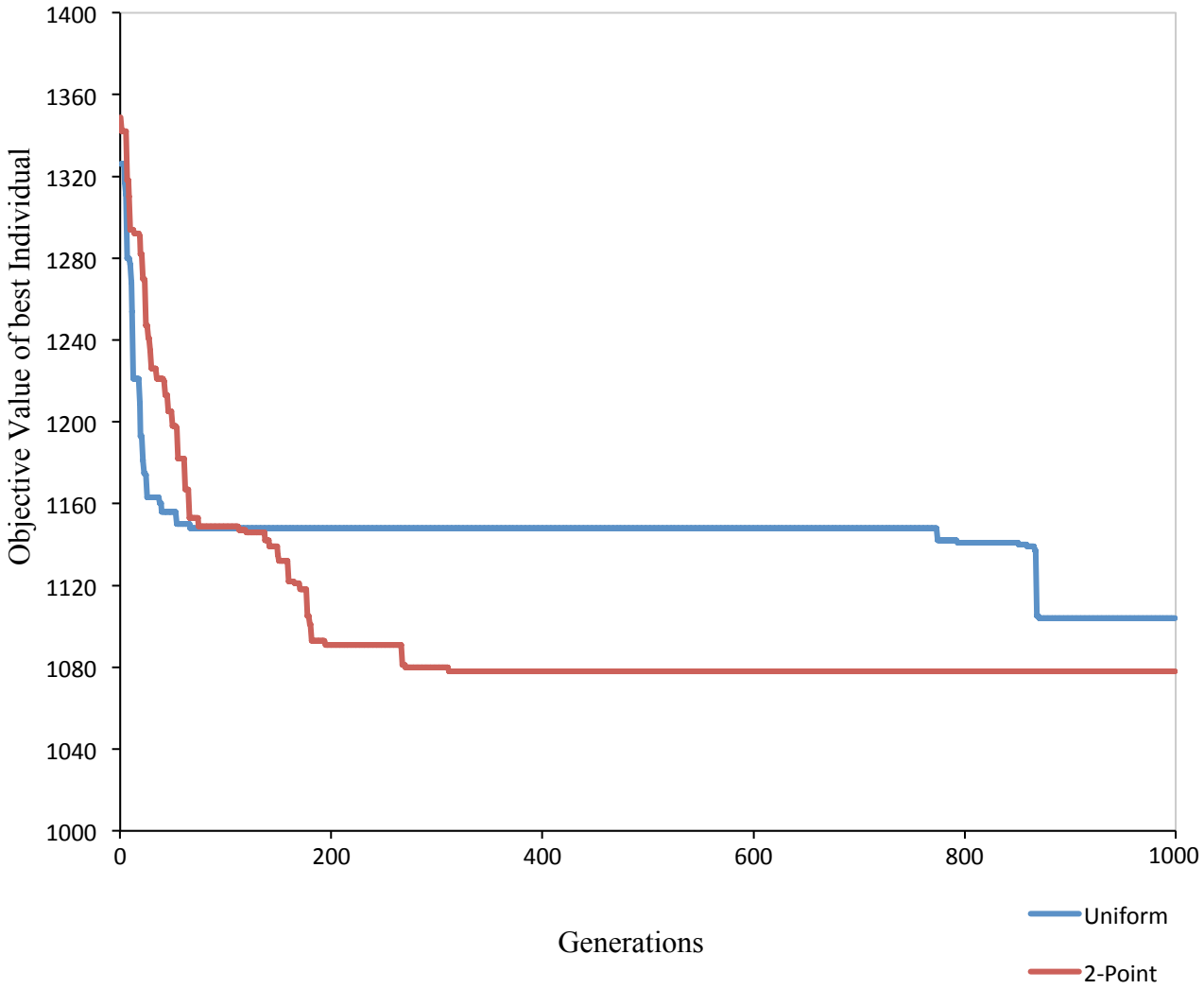


Figure 9 - The Evolution Process For Our GA Method vs. Zhang's GA Method.

The problem of premature convergence still persists. So, we were motivated by Simulated Annealing. Simulated Annealing is called probabilistic hill-climbing (Adler) which alters the inferior solutions acceptance probability dynamically. Hill Climbing always selects the better move, while Simulated Annealing (SA) selects a random move from the neighbourhood based on acceptance probability. If the move is better than its current position, then new position is always taken. If the move is worse, then new position will be accepted based on acceptance probability. This may help to bounce back without getting trapped into local minima, thereby avoiding premature convergence. So, SA Method was modeled to NPP. For Details, see Chapter 3.

SA parameters considered are set as follows: Move probability, $P_m = 0.75$, Schedule Factor, $S_f = 0.9999$, Initial Temperature, $T_{ini} = 1000$. The SA routine is executed twenty times, each time a different weight coefficient β (t) = $0.001 * 1.6^t$, $1 \leq t \leq 20$) is set for the fitness function. Table 11 displays the data collection after implementing SA routine:

Table 11 - Data Collection From SA Method

Weight Coefficient(β)	Aggregate inter-network communication(C)	Communication Imbalance(I)	Objective Value(f)	Improvement (%)
0.0016	1059	246	1059	2.216066482
0.0026	1066	326	1066	-3.294573643
0.0041	1067	187	1067	-1.522359657
0.0066	1081	293	1082	-4.138594803
0.0104	1030	183	1031	-2.180376611
0.0168	1064	357	1069	-3.585271318
0.0268	1062	305	1070	-7.214428858
0.0429	1066	188	1074	-3.768115942
0.0687	1084	177	1096	-5.486044273
0.1099	1079	83	1088	-6.146341463
0.1759	1076	130	1098	-4.571428571
0.2814	1074	74	1094	-2.819548872
0.4503	1084	89	1124	-7.149666349
0.7205	1059	94	1126	-7.034220532
1.1529	1117	17	1136	1.303214596
1.8446	1149	2	1152	-1.408450704
2.9514	1112	9	1138	10.25236593
4.7223	1156	5	1179	6.576862124
7.5557	1175	4	1205	4.743083004
12.089	1147	5	1207	11.89781022

For each weight coefficient, we have compared best individual's objective values of our SA Method and Zhang's GA Method. Column 5 of Table 11 indicates the percentage of improvement compared with Zhang's GA Method. From Table 11, we can see that the Pareto solutions, for higher values β, improved a lot through SA Method. Figure 10 is the graphical representation of Table 11. From the above data, the Pareto solution with β = 0.2814 is apparently a good selection.

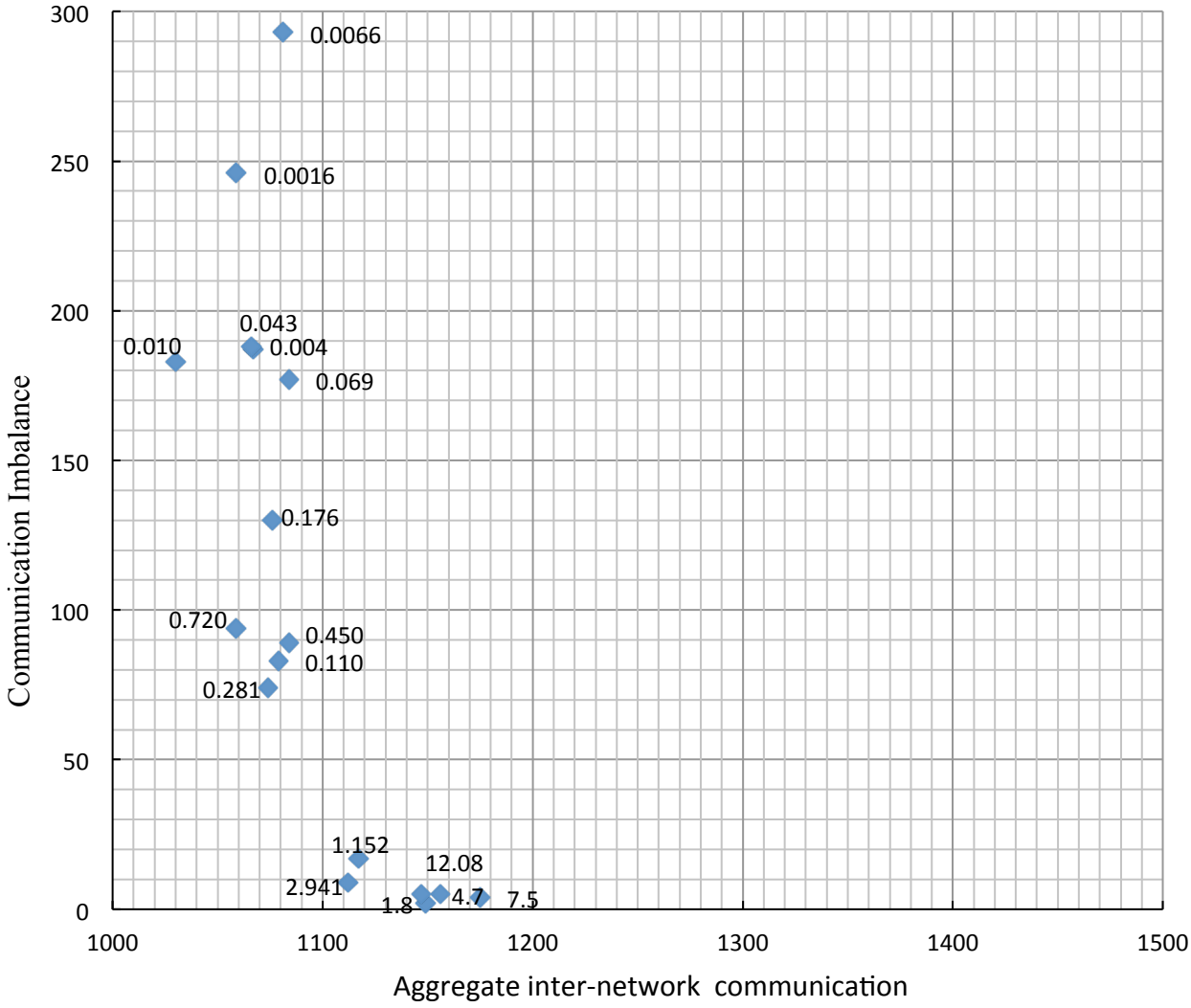


Figure 10 - The Obtained Pareto Solutions With Different Weight Coefficients, And The Pareto Solution With β=0.281 is Selected.

Based on our research, we propose that uniform crossover GA method has delayed premature convergence in NPP, thereby improving the Pareto solution. We have observed that combination of GA and SA has improved all Pareto solutions; we have tried, for different weighted coefficients. Combination of GA and SA Methods would run GA and SA routines simultaneously and select the best solution as Pareto solution for a given β. Few

issues, like difference in input parameters and runtime factors, caused our attempt to use a truly interleaved method fail. We will consider this as our future work.

CHAPTER 5

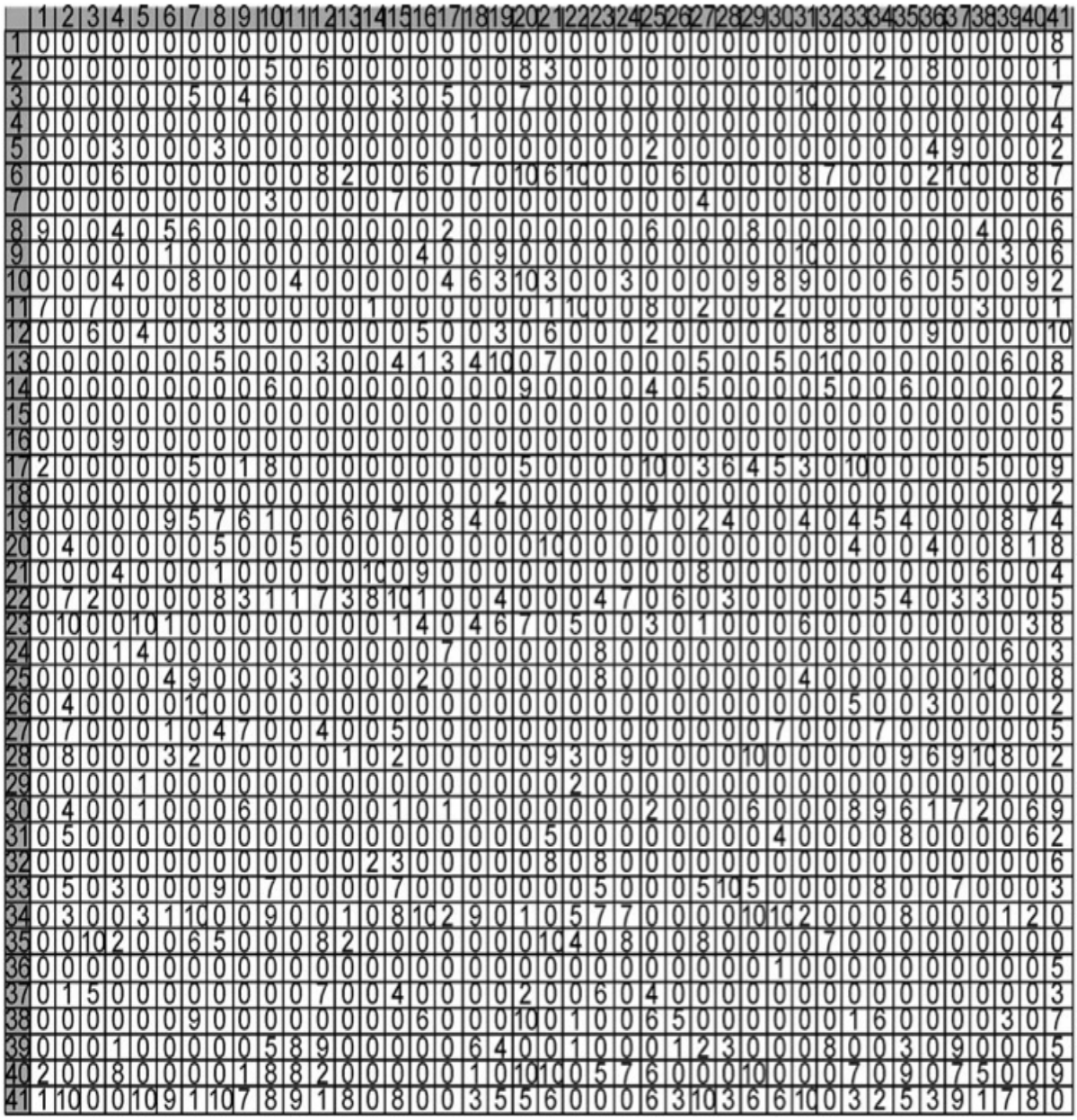
CONCLUSION AND FUTURE WORK

In this work, mathematical model proposed, in the paper "Network partition for switched industrial Ethernet using genetic algorithm" (Zhang and Zhang), for Network partition problem in switched industrial Ethernet with tree topology is corrected. Then, we have verified the results of the previous work. To avoid premature convergence in early phase of evolution process, we have proposed uniform crossover GA Method to optimize network partition. We have also proposed SA Method to optimize network partition, which seem to outperform when more emphasis is laid on minimizing communication imbalance. Our simulation results confirm that uniform crossover GA Method improved upon the problem of premature convergence and also the combination of GA and SA Methods seems effective than GA Method.

Our future research will consider designing a truly interleaved method of GA and SA. We will also consider comparing our current approach with other search methods such as neural networks and TABU search.

APPENDIX

In the appendix, the traffic matrix is given.



REFERENCES

Adler, D. "Genetic algorithms and simulated annealing: a marriage proposal." IEEE International Conference on Neural Networks. San Francisco, CA , USA, 1993. 1104 - 1109 vol.2.

Andrade, Alessandro Vivas, et al. "Analysis of Selection and Crossover Methods used by Genetic Algorithm-based Heuristic to solve the LSP Allocation Problem in MPLS Networks under Capacity Constraints." International Conference on Engineering Optimization. Rio de Janeiro, Brazil, 2008.

C., Chu, Premkumar G. and Chou H. "Digital data networks design using genetic algorithms." European Journal for Operations Research 127 (1) (2000): 140–158.

C.A., Farrell, Kieronska D. H. and Schulze M. "Genetic algorithms for network division problem." Proceedings of the First IEEE Conference on Evolutionary Computation, vol. 1. Orlando, FL, USA, 1994. 422–427.

H., Runhe, et al. "Parallel genetic algorithms for communication network design." Proceedings of Second Aizu International Symposium. Aizu-wakamatsu, Japan, 1997. 370–377.

H., Sayoud and Takahashi K. "Designing communication networks topologies using steady-state genetic algorithms." IEEE Communication Letters 5 (3) (2001): 113–115.

Hsinghua, Chou, Premkumar G. and Chu Chao-Hsien. "Genetic algorithms for communications network design—an empirical study of the factors that influence performance." IEEE Transactions on Evolutionary Computation 5 (3) (2001): 236–249.

Krommenacker, N., T. Divoux and E. Rondeau. "Using genetic algorithms to design switched Ethernet industrial networks." Proceedings of the 2002 IEEE International Symposium Vol. 1. Vandoeuvre les Nancy, France, 2002. 152–157.

M., Gen, Cheng R. and Oren S.S. "Network design techniques using adapted genetic algorithms." Advances in Engineering Software 32 (9) (2001): 731–744.

Pierre and F. Houéto. "Assigning cells to switches in cellular mobile networks: a comparative study." Computer Communication 25 (5) (2002): 464–477.

Quintero, A. and S. Pierre. "Evolutionary approach to optimize the assignment of cells to switches in personal communication networks." Computer Communication 26 (9) (2003): 927–938.

R., Elbaum and Sidi M. "Topological design of local-area networks using genetic algorithms." IEEE/ACM Transactions on Networking 4 (5) (1996): 766–778.

S., Pierre and Legault G. "A genetic algorithm for designing distributed computer network topologies." IEEE Transactions on Systems, Man and Cybernetics, Part B 28 (2) (1998): 249–258.

Salcedo-Sanz and X. Yao. A Hybrid Hopfield network-genetic algorithm approach for the terminal assignment problem. IEEE Transactions on Systems, Man and Cybernetics, Part B 34 (6), 2004.

Songerwala, M. "Efficient solutions to the network division problem." Ph. D. Thesis, Department of Computer Science, Curtin University of Technology. 1994.

Zalzala, A.M.S. and P.J. Fleming. "Genetic Algorithms in Engineering Systems." Incorporated/Institution of Electrical Engineers. USA: INSPEC, 1999.

Zhang, Qizhi and Weidong Zhang. "Network partition for switched industrial Ethernet using genetic algorithm." Engineering Applications of Artificial Intelligence 20 (1) (2007): 79-88.

VITA

Vamshidhar Reddy Boda, received B.S. in Computer engineering from Osmania University, India in July 2008. He is pursuing M.S. in Computer Science at University of Missouri Kansas City. His research focuses on applying intelligent technologies in communication networks design and evolutionary computing.