Proceedings of the 17th World Congress
The International Federation of Automatic Control
Seoul, Korea, July 6-11, 2008

NETWORK TOPOLOGY DESIGN ⋆

T. Fencl, P. Burget, J. Bilek *

* *Czech Technical University in Prague,*
Faculty of Electrical Engineering, Department of Control engineering,
Technicka 2, Prague 6, 16627, Czech Republic
(e-mail: fenclt, burgetpa, bilekj@control.felk.cvut.cz).

Abstract: The application of the Industrial Ethernet brings up issues of both the reliability and behaviour of the networks under a demanded load, while preserving the permitted time delay in the whole network. We propose an algorithm that designs the network topology, which meets the required fault tolerance and allows reliable communication between the parts of the control system. For this purpose, we employ an iterative genetic algorithm that designs the physical topology and verifies the behaviour under the expected load and time delay in the whole network. We propose the chromosome representation and genetic operators required. The network results compare physical topologies, which were acquired by a common algorithm and by our algorithm.

Keywords: Networks; Network reliability; Network topologies; Ethernet; Genetic algorithm.

1. INTRODUCTION

The demand for reliable communication grows together with the growing possibilities of distributed control systems. Unfortunately, the issue of network topology design is often omitted, and therefore, we propose an algorithm for the design of a communication network in control engineering. The resulting network not only ensures the delay in the whole network with minimal acquisition costs but also offers different degrees of reliability in different parts of the network.

Many common algorithms for network topology design are aimed towards the design of computer networks. Computer networks are used in different areas of human life, but the basic requirements are the same: a certain degree of reliability, a constrained average delay of data delivery and of course minimal acquisition costs.

Communication networks for control engineering (control networks) are applied in industry and, therefore, must satisfy much stricter constraints, e.g.: different degrees of tolerance to the links interruption in different parts of the network (due to the different probability of an interruption in the different network parts). An average delay of data delivery can hardly be used as a measure of the network's ability for real-time control systems because it does not ensure maximal delay for all data-flows in the worst case. Late delivery of data can cause serious issues (malfunction of the control system and, as a consequence, malfunction of the controlled system). Hence, the design of a control network must employ different approaches than the design of a computer network.

⋆ This paper has been co-financed by the Grant Agency of the Czech Republic, project number 102/08/1429 "Safety and Security of Networked Embedded System Applications" and further supported by the Czech Ministry of Education, Youth, and Sports as EuSophos project No.2C06010

Algorithms that ensure the average delay and the same minimum degree of reliability in the whole network were proposed (Szlachcic 2006, Ko et al. 1997) for computer networks. These algorithms were developed to be quite fast and efficient and employ genetic algorithms. We can use them if we require the same minimum degree of reliability in the whole network and the average delay of data delivery is important. Another possibility is to use algorithms that do not focus on reliability but on other attributes of network (Kumar et al. 1998, Han et al. 2002, Thompson and Bilbro 1998, Elbaum and Sidi 1995, Dutta and Mitra 1993). This possibility is not an option for us because we expect to have network applications in control engineering.

The reliability of the network is connected to the tolerance of the link interruption. The fault tolerance can be described by the network connectivity. The connectivity describes how many independent paths exist between a pair of nodes. The network (graph) is k-connected if every node in the network has k or more communication links (edges); if k-1 links are interrupted, the network is still connected and the network is k-1 fault tolerant (Cheng 1998). This attribute allows the development of a very fast algorithm because we can verify k-connectivity by finding the minimum node degree in the network. It is possible to use it in control engineering as well, if we demand the same minimum number of independent paths for each node pair in the network. However, the resulting network is unnecessarily expensive. We can imagine that the control system needs 2-connectivity in a bigger part of the system, but in the "core" of the system, we want to reach 4-connectivity. We must set 4-connectivity for the whole network in the algorithm (Szlachcic 2006, Ko, et al 1997) and, therefore, the acquisition costs of the resulting network would be higher than is necessary.

It is very difficult to satisfy the demands for the different number of independent paths between the nodes, the max-

imal delay in each communication path and the minimal acquisition costs at the same time. This task belongs to the NP-complete problems and, therefore, we use a topological iterative genetic algorithm, which ensures the lowest possible acquisition prices while preserving the reliability (e.g. the redundant communication path between the nodes).

2. PROBLEM FORMULATION

Let $G = (V, L)$ be the network, with V being the set of nodes and L being the set of communication links, which we want to gain. We need to know the other network parameters; M is the matrix of redundancy (it describes how many independent paths are demanded between each pair of nodes); C is the matrix of acquisition costs. We assume that:

$$c_{i,j} = c_{j,i}; \forall i, j \in \{1, .., N\}, \tag{1}$$

where N is the number of nodes. The parameters that must be known for verification of the delay are as follows: F - the matrix of data-flows (it describes the amount of data-flow between the nodes; we gain this matrix from an estimation of the network traffic during the design of the control algorithm. We know the variables, which every node needs to gain from the other nodes; we also know the frequency of the variables update as well as the attributes and the behaviour of the communication protocol. Then we can estimate the traffic among nodes), D - the matrix of the maximum permitted delays (we know the sensitivity of the control algorithm in every node to the late data-delivery, thanks to that we can estimate the maximum permissible delay of the data delivery); if we do not have demands on the delay, the corresponding element is infinity. We assume the capacity of the network links is given at the start of the design and it is the same for each communication link in the network. Such an assumption is realistic and puts no constraints on the applicability of the results.

We apply these variables to the design of the physical network topology (it describes how to connect the nodes with the communication links), which allows one to design logical topology (it describes how to send data between the nodes) and, therefore, makes verification of the data delay possible.

3. DESIGN OF THE PHYSICAL TOPOLOGY

Design of a network topology with respect to the aspects of connectivity and minimal network cost belongs to the NP-complete problems and we, therefore, employ a genetic algorithm for this purpose. The genetic algorithm does not ensure identification of the optimal solution, but it has a great advantage of searching very fast in a huge number of solutions and the easy application of several constraints. The speed of the genetic algorithm is given by the genetic operators: mutation and crossover. The genetic algorithm works like evolution in nature. Mutation and crossover change the actual population and selection determines which offspring or individuals of the actual generation will be in the next generation. Selection utilises chromosome evaluation that describes the quality of the chromosomes. The evaluation of the chromosomes is obtained, thanks to a fitness function.

3.1 Chromosome representation

The chromosome that describes network topology is a vector i with l elements.

$$i = \{i_j : i_j \in \{0, 1\}, j = 1, .., l, \tag{2}$$

l is given by

$$l = \frac{N(N-1)}{2}, \tag{3}$$

where N is the number of nodes in the network. Vector i characterises the upper triangular part of the adjacency matrix P because the adjacency matrix is symmetrical. The fact that P is symmetrical, arises from the assumption of a full-duplex communication where one communication link permits communication in both directions.

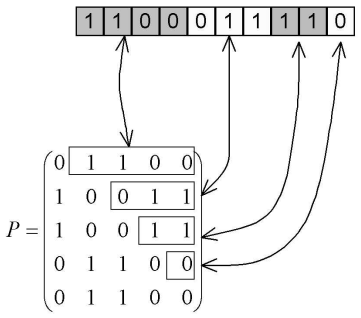

Fig. 1. Adjacency matrix

3.2 Genetic functions

The fitness function for the design of the physical topology is the price of the whole network. The price includes all acquisition costs and possible penalties. The price of the network is the sum of all elements of matrix C_P:

$$C_P = CP. \tag{4}$$

The final price of the physical topology is given by:

$$c = \sum_{i,j=1}^{N} c_{P_{i,j}} + Pen, \tag{5}$$

where N is the number of nodes and $c_{P_{i,j}}$ are elements of the C_P. Then the penalty is:

$$Pen = 1 + N^2 c_{MAX}, \tag{6}$$

where c_{MAX} is the maximum element of the matrix of the acquisition costs. Equation (6) ensures that the chromosome, which is penalised, has a bigger price than the permissible chromosomes. The chromosomes are penalised if they do not correspond to the demanded number of redundant communication links or if they are the same as one of former unsuitable solutions. Former unsuitable solutions are stored in an accumulator of inappropriate topologies. The inappropriate topologies are such topologies in which the delay is higher than the permitted value. We obtained those solutions in the previous iterations of our algorithm (the whole algorithm is described at the end of this paper). We need to verify the network interconnection and the number of independent paths between the nodes

before applying the penalty. We apply the Ford-Fulkerson algorithm (Cormen, et al 2001) to verify the number of independent paths and the degree of nodes to make sure they are interconnected.

The network is connected if:

$$\deg(\nu) \geq 1; \forall \nu \in V \ \wedge \sum_{\forall \nu \in V} \deg(\nu) \geq N - 1. \qquad (7)$$

We apply a one-bit mutation (a mutated chromosome is chosen at random according to the mutation probability and a bit is also chosen at random) and one point crossover (two chromosomes are crossed at the same place, the offspring consist of part of both parents as depicted in Fig. 2). The new offspring replace the parents only if they have better quality.

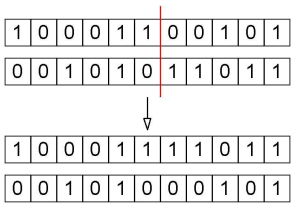

Fig. 2. Crossover

We use tournament selection (two chromosomes selected at random are compared and the better one is chosen for the next generation). The genetic algorithm is stopped after 100 generations.

4. AVERAGE BIT DELAY

The attributes of a network in control engineering are the same as the attributes of a computer or communication network. The network has Poisson distribution at the inter-arrival time and the service time as well. Then the average delay of the next sent bit is

$$D_p = \frac{L_p}{C_{ap} - L_p}, \qquad (8)$$

where C_{ap} is the link capacity and L_p is the load of the link. Then the maximal load for the permitted delay is:

$$L_p = \frac{D_p C_{ap}}{D_p + 1} \qquad (9)$$

We can transform the matrix D (the matrix of the maximum permitted delay) into the matrix of the maximum incoming data-flows $F^{\bullet}$. Then, we can transform matrix $F^{\bullet}$ into a vector of the maximum incoming data-flows into the node easily:

$$\bar{f} = (f_1^{\bullet} \ \ldots \ f_n^{\bullet}), \qquad (10)$$

where

$$f_i^{\bullet} = \min\left(F_{x,i}^{\bullet}\right) \quad for \ \forall i, x \in \langle 1, N \rangle. \qquad (11)$$

We apply (10) to decide whether the resultant logical topology meets the demands for the average delay.

5. DESIGN OF THE LOGICAL TOPOLOGY

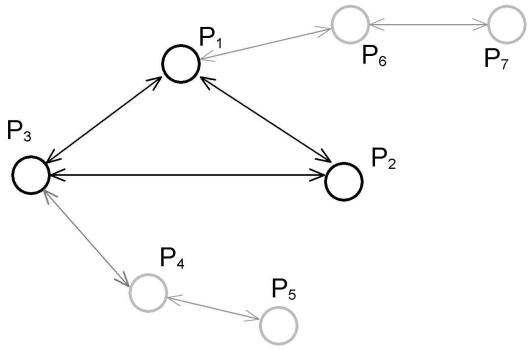

Fig. 3. Network

Fig. 3 shows a possible physical topology that meets the demand for two independent paths between pairs of nodes P_1, P_2 and P_1, P_3. The network consists of two different parts: branches of the network (P_1,P_6,P_7; P_3,P_4,P_5) and the "core" of the network (P_1,P_2,P_3). The logical topology in the branches is strictly given by the physical topology and, therefore, we do not have any possibility to change the logical topology without modification of the physical topology. We can find the branches of the network thanks to the degree of nodes. We find the start node of the branch (it has $\deg(\nu) = 1$), and then every node, which is connected to the start node and has a degree of less than 3, is a component of the branch. The end node of the branch has the degree $\deg(\nu) \geq 3$. We find all branches and exclude them from the logical topology design. (We do not have any influence on the logical topology in the branches). Before continuing the design, it is appropriate to verify the delay in the branches (the sum of all the incoming data-flows into each node must be smaller than the corresponding element in the vector (10)), because it is possible that the physical topology does not allow one to keep the permitted delay in the branches and, therefore, it is necessary to start the design of the physical topology again. Excluding the branches decreases the number of possible logical topologies and, therefore, increases the algorithm speed. We can see it in the following example: The original data-flows for our network looked like this:

$$F = \begin{pmatrix} 0 & f_{1,2} & f_{1,3} & f_{1,4} & 0 & 0 & f_{1,7} \\ f_{2,1} & 0 & f_{2,3} & 0 & f_{2,5} & f_{2,6} & 0 \\ f_{3,1} & f_{3,2} & 0 & f_{3,4} & 0 & 0 & 0 \\ f_{4,1} & 0 & 0 & 0 & f_{4,5} & 0 & f_{4,7} \\ 0 & f_{5,2} & 0 & f_{5,4} & 0 & f_{5,6} & 0 \\ 0 & f_{6,2} & 0 & 0 & f_{6,5} & 0 & f_{6,7} \\ f_{7,1} & 0 & 0 & f_{7,4} & 0 & f_{7,6} & 0 \end{pmatrix}. \qquad (12)$$

The matrix (12) is changed to the (13) after removing branches

$$F_M = \begin{pmatrix} 0 & f_{M,1,2} & f_{M,1,3} \\ f_{M,2,1} & 0 & f_{M,2,3} \\ f_{M,3,1} & f_{M,3,2} & 0 \end{pmatrix}, \qquad (13)$$

where the elements of the matrix F_M comprise the following data-flows: In a new matrix (13) are not only data-flows with input and output node in the "core" but also data-flows with input or output outside of the "core". (Data-flow which is going through the "core" has new

input and output nodes. They are the first and last node in the "core" for that data-flow).

$$\begin{array}{l} f_{M,1,2} = f_{1,2} + f_{6,2} \\ f_{M,1,3} = f_{1,3} + f_{6,5} + f_{7,4} \\ f_{M,3,1} = f_{3,1} + f_{4,1} + f_{4,7} + f_{5,6} \\ f_{M,3,2} = f_{3,2} + f_{5,2} \end{array}. \quad (14)$$

We must presume a smaller capacity of communication links because if some communication link is interrupted, its load will be sent through the other links. If the substitute links are fully utilised, the data will be delivered too late; therefore, we can only count on a smaller link capacity that is given by:

$$k = 1 - \frac{1}{n}, \quad (15)$$

where n is the number of independent paths; then the new capacity in the "core" of the network is:

$$C_{PN} = kC_{ap}. \quad (16)$$

Then we use C_{PN} instead of C_{ap} in (8) to calculate the delay in the network core.

5.1 Logical topology representation

The chromosome for the logical topology description codes every possible path for every data-flow in the "core" of the network. The length of the chromosome is given by:

$$l = jm, \quad (17)$$

where j is the number of the data-flows and m is the number of communication links in the "core" of the network. In our case, $j = 11$ and $m = 3$ (see (14), Fig. 3) and, therefore, the length of chromosome is $l = 33$.

5.2 Genetic operators

We have to modify the genetic operators for our chromosome configuration. The crossover function can be modified very easily; we know that a chromosome comprises groups of bits that represent the logical path for every data-flow. Therefore, our one-point crossover function crosses the chromosomes exactly at the borders of these groups. If the chromosomes are crossed somewhere else, the resultant logical topology can be impermissible. We can see the possible crossing point in Fig. 4.

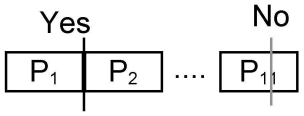

Fig. 4. Crossover function

In our case the logical paths P_x, which describe every data-flow, are comprised of 3 bits. The LSB (Least Significant Bit) corresponds to the link between nodes P_1 and P_2. The MSB (Most Significant Bit) corresponds to the link between nodes P_3 and P_1.

We have to use our own mutation operator for the description of the logical topology. If we used the common one-bit mutation for our problem, the results obtained by this operator would be completely wrong because they would represent an impermissible logical topology.

We show a mutation application for the data-flow $f_{1,2}$. First, we have to create a list of all possible logical paths for each data-flow. The possible paths are represented by only two binary chains, these being 001 and 110 (the coding of the path is the same as for the crossover). If the chain 001 is mutated, its value will be transformed into chain 110 and vice versa. We have to expect that there are more than two possible binary chains. In this case, the mutation result is chosen at random from the set of the remaining paths. The lists of the possible paths are also created during the chromosomes initialisation. Initialisation is completed by setting a randomly chosen logical path for each data-flow.

5.3 Fitness function

An important part of the genetic algorithm is the fitness function and we have to define it for our problem. We can calculate the delay in the network very easily because each chromosome exactly defines the whole logical topology for the "core" and the rest of the logical topology (the logical topology in branches) is known from the past. We can use the average delay or the sum of delays in every link as the fitness function. If the average delay decreases, then the quality of the logical topology will increase.

It is possible that the average delay is satisfactory but the delay in one path is bigger than the permitted delay and, therefore, the whole logical topology is wrong. We have to be able to recognise some level of inappropriate designs and, therefore, we add a penalty to the fitness function. A different penalty is needed for each rank of inappropriate chromosome according to the number of delays that are larger than permitted delays.

$$Penalty = k \max(D), \quad (18)$$

$$Fit = \sum_{\mathrm{i}=1}^{\mathrm{j}} \mathrm{d_i} + Penalty. \quad (19)$$

Equation (19) is our fitness function that describes the quality of chromosome; d_I is the delay for the i^{th} data-flow and it is obtained from (8) for every data-flow in the "core". Equation (18) describes the penalty function; k is the number of delays that are bigger than the permitted delay.

5.4 Selection and end condition

We use a tournament selection because of its efficiency. We know the best chromosome has the smallest delay (the best quality).

Our end condition is the speed of improvement in last ten iterations of the genetic algorithm. The algorithm will be stopped if the improvement in the last ten iterations is smaller than 0.5%.

6. ALGORITHM DESCRIPTION

As mentioned above, the whole algorithm is an iterative task based on the genetic algorithm. We can see the structure of the whole algorithm in Fig. 5.

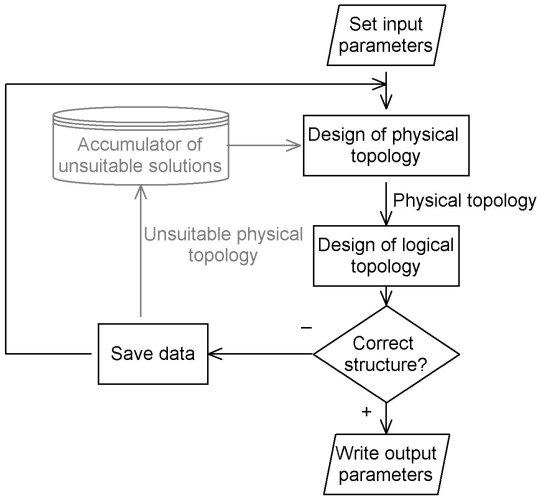

Fig. 5. Whole algorithm

Set input parameters The matrices of the data-flows, permitted delays, acquisition costs and redundancy are set in this step. The accumulator of inappropriate solutions is cleared.

Physical topology The physical topology design is obtained by the application of the genetic algorithm in this step. This genetic algorithm is a common genetic algorithm that uses representation, genetic operators and the fitness function described in Section 3. The designed topology is compared with the topologies stored in the accumulator of inappropriate topologies. The resuling topology is used as the input for the design of the logical topology.

Logical topology In this step, it is verified whether the physical topology allows the maximum permitted delays to be preserved for each data-flow. For this purpose, a logical topology is designed. If the physical topology does not meet the demands for the permitted delay, the physical topology is stored in the accumulator of inappropriate topologies and the algorithm starts to design the physical topology again. If the network allows the maximum permitted delay to be preserved, the algorithm ends and the results are written to the output.

We can see that it is possible that our algorithm could iterate to infinity because of unrealistic demands. Therefore, it is necessary to set the number of iterations of the whole algorithm. If the limit number of iterations is exceeded, the algorithm is stopped and the best result from the accumulator of the inappropriate solutions is written to the output.

7. NUMERICAL RESULTS

The tests of the physical topology design were made for medium sized networks with different numbers of nodes ($N = 10, 12, 14, 16, 18, 20$). The experiments were done with the following parameters: population size 200, crossover probability $p_c = 0.25$, mutation probability $p_m = 0.05$, the number of iterations 100. The PC was a Pentium 4 - 3GHz, 512MB RAM, Windows XP-SP2. The algorithm was implemented in C#.

N	$k=3$ $T[ms]$	$k=4$ $T[ms]$	$k=5$ $T[ms]$
10	118	120	121
12	170	172	172
14	224	223	224
16	288	288	287
18	356	357	356
20	439	440	440

Table 1. Time consumption for k-connectivity

We can see that the algorithm is not sensitive to the requested number of k-connectivity, but is sensitive to the number of nodes. This attribute of the algorithm is expected because when the node number increases the length of the chromosome rises as well. Moreover, when the length of the chromosome is bigger then the chromosome creation needs more time and, therefore, the algorithm is sensitive to the number of nodes.

N	j	$T[s]$
10	4	2.86
12	5	4.41
14	6	6.49
16	7	9.39
18	8	13.42
20	9	18.62

Table 2. Time consumption for the different number of redundant paths

In Tables 2 and 4 j represents the number of node pairs, for which 3 independent interconnecting paths are requested. For the remaining pairs of nodes only two independent paths are requested. We can see that the algorithm needs a longer solution time but this behaviour is natural, because the algorithm has to verify the number of independent paths.

N	μ	σ	min
10	15.56	0.05	15
12	18.96	0.06	18
14	22.07	0.06	21
16	25.55	0.09	24
18	28.85	0.09	27
20	32.66	0.1	31

Table 3. k-connectivity, statistical results

N	j	μ	σ	min
10	4	13.41	0.05	13
12	5	16.42	0.08	15
14	6	19.38	0.08	18
16	7	22.11	0.09	20
18	8	25.27	0.09	23
20	9	28.63	0.11	26

Table 4. Physical Topology design, statistical results

In tables 3 and 4 min is the minimal value of the found acquisition costs, μ is the average acquisition cost and σ is the variance. These results were obtained from 100 runs of the network topology design algorithm. Tables 3 and 4 contain the results for a network in which all the pairs excluding the "core" should have at least two independent

paths between the nodes. There should be three or more independent paths between the node pairs in the "core". The "core" contains the minimal number of nodes according to j^{th} column in Table 4. Table 3. describes the results, which were gained by the common algorithm for 3-connectivity in the whole network. (This setting ensures that there are three independent paths for each pair in the "core"). Table 4 describes the results, which were gained by our algorithm for the requested number of independent paths. The results in Table 3. and 4. were obtained from runs for the same set of input parameters. The acquisition costs for each link were set to 1. Thanks to that, we can compare the acquisition costs of the resultant network. We can see that the average costs and the minimum costs in Table 4. are in all cases smaller than in Table 3. According to these results, we can conclude that our algorithm designs a better physical topology with the requested number of independent paths in a reasonable time.

8. TIME CONSUMPTION

8.1 Design of the physical topology

$$\Theta_P = N_{PCh}\left(\frac{N\,(N-1)}{2} + Ln\right) + \\ + \left(Ln\,N_{PCh}\,(p_{Pc} + p_{Pm}) + 3N_{PCh}\right)N_P. \quad (20)$$

Where the first part belongs to the initialization of the gen. algorithm, L is the number of com. links, N is the number of nodes, n is the number of independent com. path, N_{PCh} is the number of the chromosome. The second part belongs to the iterations of the gen. algorithm p_{Pc} and p_{Pm} are the probability of the crossover and the mutation for the design of the physical topology, 3 belongs to the number of operations which are necessary for the tournament selection and N_P is the number of the iteration of the algorithm for the physical topology design.

8.2 Design of the logical topology

$$\Theta_L = N^3 + LnN_F + (N_{LCh}N_F)\,4LN_F + \\ +(N_{LCh}\,(p_{LC} + p_{LM})\,4LN_F + 3N_{LCh})N_L \quad (21)$$

Where N_F is the number of data-flow, N_{LCh} is the number of chromosomes,p_{Lc} and p_{Lm} are the probabilities of the crossover and the mutation, respectively, and $N_L$is the number of iterations.

8.3 Computing time of the whole algorithm

Then the time complexity of the whole algorithm is

$$\Theta = N_A\,(\Theta_L + \Theta_P)\,, \quad (22)$$

where N_A is the number of iterations of the whole algorithm after which the algorithm is stopped without finding an appropriate solution (demands are unrealistic). Equation (22) describes the worst case that could happen. This situation arises from the unrealistic demands for permitted delays which are not possible to reach because of the other demands on the network.

9. CONCLUSION

The numerical results that were described in Section 7, show that the topology gained by our algorithm has a smaller acquisition cost than the topology gained by existing algorithms (the algorithm that uses only k-connectivity as the test of fault-tolerance); the variance of the results for both algorithms is quite similar. Networks designed by the existing algorithm always have bigger acquisition costs because of the principle applied. It is obvious that the existing algorithm, which uses k-connectivity, design a more expensive network than our iterative algorithm, when we need network topology with different degrees of reliability. The cost savings is more than 10% and these results prove that our algorithm designs networks with smaller acquisition costs. The obtained physical topology is used as the input to the second part of our algorithm, which was described in Section 5. The whole algorithm provides a design of a reliable inexpensive network that can be used in control engineering because the network is fault tolerant and allows data delivery to the destination in the time demanded.

REFERENCES

Szlachcic E. *Fault tolerant topological design for computer network.* Proceedings of the International Conference on Dependability of Computer Systemes, 2006.

Han J., Malan G. Robert , Jahanian F.. *Fault-Tolerant Virtual Private Networks within An Autonomous System.* Proceedings of the 21 st IEEE Symposium on Reliable Distributed Systemes, 2002.

Cormen T.H., Leiserson Ch.E., Rivest R.L., Stein C.. *Introduction to Algorithms.* In A.F. Round, editor, McGraw-Hill, New York, 2nd edition, 2001.

Kumar G., N. Narang, C.P. Ravikumar. *Efficient Algorithms for Delay-bounded Minimum Cost Path Problem in Communication Network.* Proceedings of the Fifth International Conference on High Performance Computing. ISBN:0-8186-9194-8, 1998.

Thompson D. R., Bilbro G. L. *Comparison of Two Swap Heuristics with a Genetic Algorithm for the Design of an ATM Network.* Proceedings of the International Conference on Computer Communications and Networks, 1998.

Cheng S. T. Topological Optimization of a reliable Communication network *IEEE Trans. Reliability*, vol. 47, No. 3 pages 225–233, 1998.

Ko King-Tim, Tang Kit-Sang, Chan Cheung-Yau, Man Kim-Fung , Kong Sam. Using Genetic Algorithms to Design Mesh Network. *IComputer*, vol. 30, No. 8 pages 56–61, 1997.

Elbaum R., M. Sidi *Topological design of local area network using genetic algorithms.* Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies, 1995.

Dutta A., S. Mitra Integrating Heuristic Knowledge and Optimization Models for Communication Network Design. *IEEE Transactions of knowledge and data engineering.*,vol 5,pages 999-1017 Proceedings of the Fourteenth Annual Joint Conference of the IEEE Computer and Communication Societies, 1993.