Czech Technical University in Prague
Faculty of Electrical Engineering
Department of Control Engineering

Algorithm for network topology design

Doctoral Thesis

Tomáš Fencl

Prague, March 2011

Ph.D. Programme: Electrical Engineering and Information Technology
Branch of study: Control Engineering and Robotics

Supervisor: ***Doc. Ing. Jan Bílek CSc.***

Acknowledgements

In the first place, I would like to thank my advisor Jan Bílek for his support and patience when not everything was going well and for his human quality. I would like to thank also Pavel Burget for his support and help. My big thanks belong to my family and friends who supported me in every possible way during those years and make me smiled when I needed it. I would like to thank to those all who I met during this time and allowed to me see things from different points of view and help to recognize what is really important. My thanks belong also to my colleagues for friendly atmosphere. I want to say thanks also to all people who have read all papers and works before its submitting and advised me how to improve it. My thanks belongs to all anonymous reviewers for their comments that allows to improve this work.

This works 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 by the Czech Ministry of Education, Youth, and Sports as Eu-Sophos project No.2C06010 This work was supported by the Ministry of Education of the Czech Republic under project 1M0567.

Czech Technical University in Prague *Tomáš Fencl*
March 2011

Nomenclature

C	Matrix of the acquisition costs; $c_{i,j}$ describes the costs that is necessary to pay for the connection of the nodes i and j
C_F	Matrix of the costs for the fibre cable
C_M	Matrix of the costs for the metallic cable
cap_i	Capacity of the communication link
$Cost$	Costs of the designed network as well as the fitness function
D	Matrix of the maximal permitted delays (a worst-case scenario)
$Depth$	Depth of a node in the tree topology
$Depth_{max}$	Maximal permitted depth of the tree topology
e_r	Number of links connected to the nodes that should be removed
F	Matrix of data-flows, $f_{i,j}$ - size if the data-flow between nodes i and j
j	Number of data-flows in the network core
K	Array describing the number and kind of communication ports, k_{fi} - the number of communication ports of node i that can be connected to the fibre cable, k_{mi} - the number of communication ports of node i that can be connected to the metallic cable,
k_{dis}	Number of nodes that are disconnected from the main node (root node)
k_{cyc}	Number of cycles in the network
k	Number of nodes at which the number of communication ports is smaller than a number of available communication ports
k_D	Number of data-flows that have bigger delay than it is the maximal permitted delay
k_F	Number of independent paths
l	Length of the chromosome of the logical topology
M	Matrix of the fault-tolerance. $m_{i,j}$ - describes the number of the independent communication paths between nodes i and j
m	Number of communication links in the core
N_p	Number of iterations of the genetic algorithm for the design of the physical topology
N_{ch}	Number of chromosomes of the genetic algorithm for the design of a physical topology
N_{Lch}	Number of chromosomes of the genetic algorithm for the design of a logical topology
P	Matrix describes the designed network, $p_{i,j}$ describes the connection of the nodes i, j; 0-there is no connection, 1-there is a connection created by the metallic cable, 2 -there is a connection created by the fibre cable

$Pack_{len}$	Length of data packets
Pen	Penalty function
r	Flag of the fault-tolerance; 1-network is not fault-tolerant at the demanded level, 0-network is fault-tolerant
R_{ed}	List of the nodes to be removed
$\sum c_{e^+}$	Costs of all edges that were added to the original topology
$\sum c_{e^-}$	Costs that is necessary to pay for the removal of the communication links from the original network
u	$u = 1$ if the network described by the chromosome is already in the accumulator of the unsuitable topologies; $u = 0$ if the chromosome is not in the accumulator of unsuitable topologies

Algorithm for network topology design

Ing. Tomáš Fencl
Czech Technical University in Prague, 2011
Thesis Advisor: Doc. Ing. Jan Bílek CSc.

The application of the distributed control systems became a standard solution in the area of industrial control systems. The ability of the industrial control system is not only dependent on the ability of every part of the control system but also on their ability to communicate among them. The operation of the control system can be very limited if a part of the control system does not receive the demanded data or receives data too late. Unfortunately, the design of the suitable network topology is often omitted and the limitation of the network topology can cause the limited functionality of the control system.

The existing algorithms for the network topology design do not offer a design of the network with demanded attributes. These algorithms often do not apply some of the real life limitations such as the limited number of communication ports, fault-tolerance of the network or limitation caused by the environment in which the network will be used. Moreover, some algorithms are not able to ensure some ability that should be fulfilled (Algorithms are not able to ensure the design of the fault-tolerant network even if they offer it). Therefore, a novel algorithm for the design of the network topology is proposed in the thesis.

The algorithm is able to design the network with the different levels of fault-tolerance in the different parts of the network. The designed network is less expensive than the network designed by other existing algorithms. Moreover, the algorithm is able to work with the nodes that have different numbers of communication ports and addition of communication ports to nodes is not possible. The algorithm uses information, which can be received from the application engineers, about unsuitable structures of the network topology and is able to avoid designing of the network that contains these unsuitable structures. The proposed algorithm contains the part that verifies network ability to transfer every data-flow in demanded time.

It is often necessary to change the network topology if the controlled technology is changed. Thus, the modification of the proposed algorithm for the network topology design is described in the thesis. The redesigned network meets all demands for the fault-tolerance, application of nodes with the limited number of communication ports etc. as a brand new network. On the contrary, the redesigned network is less expensive than the new network since it uses as a big part of the original network as possible. Moreover, the modification for the tree topology was proposed.

The algorithm is designed as a modular iterative task on the basis of the genetic algorithm. The modularity allows application of the more accurate model of the network behaviour if it is needed. Moreover, it is possible to design the topology of the pipeline or electrical grid if the module for the verification of the network delays is substituted with model desribing behaviour of electrical grid or pipelines.

Goals and Objectives

The goals of this work were set as follows:

1. To design an algorithm for the design of the network with the different fault-tolerance in different parts of the network while the network allows in time data delivery.
2. Modifications of the basic algorithm for the limited number of communication ports of nodes (different physical layers can be applied) and an application of a priori information about unsuitable topologies.
3. To design an algorithm for the mesh network expansion and reduction.
4. To design an algorithm for the network topology design. The tree network has limited depth and use nodes with the limited number of communication ports of the different type of physical layers.
5. To design an algorithm for the tree topology expansion or reduction.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The networked systems gain bigger importance with the growing application of the distributed systems generally and distributed control or telecommunication system particularly. In these systems, the networks have a big importance but the proper design of them is unfortunately very often omitted. The design of the network is done very often ad hoc, in order to create the network as cheap as possible. This approach can cause the network malfunctions that can bring big financial losses or casualties in the worst-case scenario. It is easy to imagine that the network is used in the chemical industry and malfunction of the network causes the malfunction of the controlled technology. The malfunction of the controlled technology e.g. petrol distillation can induce technology explosion and ecological accident can occur. Moreover, many existing algorithms [3], [4], [5], [6], [7] for the network topology design are dedicated to some purpose and omit some of the real life limitations as the number of communication ports of the communication nodes or limitation caused by the environment in which the network is used.

Existing algorithms often use the reliability of the network as an attribute for the network design. The reliability is not good characteristic of the network behaviour in some cases especially, if the network transports the life critical data. The network can have such a topology that the reliability does not describe the behaviour of the network. In these cases, the fault-tolerance is a better characteristic of the network behaviour since it says how many communication links can be disrupted without an influence on the network operation. The applied level of the fault-tolerance ensures that user of the network can be sure that a certain disruption of the communication links does not cause the malfunction of the network. The fault-tolerance is easier to imagine for users and designers of the network and therefore it is easier to set the demands of the network fault-tolerance.

On the other hand, the required level of the fault-tolerance is necessary to choose very carefully. The level of the fault-tolerance is directly connected with the acquisition costs of the network since the bigger level of the fault-tolerance the more communication links are needed and the bigger acquisition costs are. Fortunately, in many cases it is possible to avoid this attribute since it is possible to find parts of the network that need bigger level of the fault-tolerance and parts of the network that need smaller level of the fault-tolerance. If the algorithm for the network topology design is able to design the network with the different levels of the fault-tolerance in the different parts of the network, then it is possible to avoid unnecessary costs as well.

There are more demands for the network abilities than request for the different fault-tolerance in the different parts of the network. The network must allow to in time delivery of the data. If the data are received too late, it can cause serious issue for the control algorithm and consequently for the controlled technology. Therefore, it is necessary to verify whether the network is able to transport the expected load in the demanded time. It is necessary to accommodate nodes with the different abilities in the network communication. Every node can have different number of communication ports and even if this situation occurs, the network must meet the request for the different levels of the fault-tolerance in the different parts of the network. It means that the design algorithm cannot plan to use more communication ports than there are available at the nodes.

The application engineer often knows information about the environment in which the network will be used. The environment can limit the structure of the network since it can prevent a usage of the direct connection between certain nodes. In this case, it is necessary to design the network in a different manner and connect these nodes via other nodes. The algorithm for the network topology design must use a priori information about unsuitable topologies and design the network according to this information in order to find the different topology. The network can be combined from the different communication technologies and therefore it is not possible to focus only on one communication technology; the design algorithm must be able to work with the different communication technologies at the same time.

Unfortunately, there is not such an algorithm that is able to design the network with the different fault-tolerance levels in the different parts of the network with a usage of nodes with the limited number of the communication ports and application of a priori information while the network is able to transport expected load in the requested time. Existing algorithms are able to design the network with the different levels of the reliability [3], [8], some of them are able to work with the limited number of the communication ports [9]. None is able to use a priori information about the unsuitable topologies. Some algorithms that are able to design the "fault-tolerant" network with the same level of the fault-tolerance are in fact not able to ensure that the network is really fault-tolerant [1], [2]. In certain situation these algorithms design networks that seem to be fault-tolerant at certain level but this assumption is wrong (see chapter: 2 **Reliability**).

Therefore, the algorithm that is able to design the network with the different fault-tolerance in the different parts of the network with the application of the nodes with the limited number of communication ports and use a priori information about unsuitable topologies is proposed in this thesis. Naturally, the network must be able to transfer the expected load.

The designed network can be changed during its life cycle. The changes can vary from a minor change in amount of the load to the change of the part of the topology or an expansion or a reduction of the number of the communication nodes. The algorithm for the network design should be able to redesign the network in this case. Existing algorithms [3], [10] are able to redesign the network topology only if the number of nodes is still the same but they are not able to redesign the network if the number of the communication nodes is changed. Moreover, the changed network should still meet the demands for the different levels of the fault-tolerance in the different parts of the network that uses communication nodes with the limited number of communication ports. The network must be still designed according to a priori information about the unsuitable topologies and must be able to transport expected load. Any known algorithm does not offer this possibility and therefore the new algorithm for this purpose is described in this thesis as well.

The tree topology is very often used in the industry even if it is not able to ensure the network fault-tolerance. On the other hand, the tree topology allows designing the network with as less expense as possible. There are algorithms that allow to design the network with the limited depth of the network and with the nodes with the different numbers of communication ports. Nevertheless, there is not an algorithm that is able to design the network with the tree topology with the limited depth and application of the nodes with the different numbers of communication ports and a priori information about unsuitable topologies. This algorithm is described in this thesis. Moreover, the algorithm allows using different physical layers.

It is often necessary to change the topology of the network if the control technology is changed. In this case, it is possible to create the brand new network or change the old network. Both solutions must meet the requirement for the maximal depth of the network and must be able to use nodes with the different numbers of communication ports. Moreover, the network topology must be different from those unsuitable topologies that are known a priori. The redesigned network should be as less expensive as possible. Thus, the new tree topology should use as much components of the original network as possible. It is only possibility to design as inexpensive changes as possible if the tree topology is redesigned. The algorithm that allows redesigning the network with the tree topology is described in this thesis. The algorithm uses as a big part of the original network as possible. The resultant network has a limited depth; contains nodes with the different numbers of the communication ports, and it is different from those unsuitable topologies that are known before the design of the changes of the network.

1.2 Objectives and outlines

Design of the network topology is the important part of the design of the distributed control systems. Unfortunately, the network is very often designed ad hoc or some important attributes are omitted during the design. This omission can cause that the designed network is not possible to construct since the nodes do not have enough communication ports or the environment does not allow the construction of the network of the designed structure. Existing algorithms are not able to design the network that meets all requests such as the different fault-tolerance in the different parts of the network, in time data delivery and limitation caused by the environment and attributes of the nodes at the same time. The algorithm that complies with all requirements is proposed in this thesis there.

The ad hoc design is not a big issue if the network is newly designed but in case that the network is incrementally extended or reduced, the network can lost its attributes and costs for these incremental changes can be unnecessary big even if the network does not meet all requests. The existing algorithms are able to add or remove communication links but they are not able to design changes in the network structure if the number of nodes must be changed. Therefore, the algorithm that is able to work with the changes of the number of nodes is proposed in this thesis. The changed network must meet the same demands as the original network. Thus, the network must have requested fault-tolerance, contain the nodes with the limited number of communication ports and be different from topologies that are unsuitable from some reasons (mostly because of the environment limitations).

The topology that is very often used in the control engineering is a tree topology. The tree topology has an advantage in easy expansion and small costs in comparison with the mesh topology. Nevertheless, there are some limitations in the structure of the network e.g. the number of communication ports of nodes, a depth of the tree topology (it has a direct influence on the network behaviour such as time of the data delivering and network behaviour if some communication link is interrupted) and limitation caused by the environment. Any existing algorithm does not apply the limitation caused by the environment. Therefore, this algorithm is proposed in this thesis.

The changes of the structure of the tree topology are very often while the changes in the controlled technology occur. These changes are mostly done ad hoc and can decrease the quality of the network (some branches of the tree can be too long). Therefore, the algorithm that is able to design as inexpensive changes in the network structure as possible is proposed in this thesis. The newly designed network meets all requests arose from the owner of the network (such as the maximal depth of the network, limitation of the number of communication ports of nodes and the limitation caused by the environment).

The genetic algorithm is used for the design of the network topology. The algorithm is time demanding for the design of the middle and bigger sized network (10+).

Therefore, a parallelization of the genetic algorithm is proposed in this thesis. The parallelization is used for the finding of the recommended setting of the algorithm for the design of the network with the different levels of the fault-tolerance in the different parts of the network.

The document is organized as follows: the first chapter defines the requests that the network should achieved and reasons why the appropriate network is so important. In the first chapter, there is basic general information about the genetic algorithm since the genetic algorithm is used for the design of the network topology.

The second chapter is dedicated to a description of the reliability and the fault-tolerance of the network. There are examples that represent differences between reliability and fault-tolerance and how important is understanding to these differences. There is also an example of misunderstanding of the meaning of the fault-tolerance approximation by the node degree and it is possible to see the result of this misunderstanding. This misunderstanding can cause that the network that should be fault-tolerant according to the approximation is not fault-tolerant in fact.

The third chapter is an overview of the parts of works related to the subject of the designs of the reliable/fault-tolerant networks. There are examples of works that should cover all possibilities of designing of the fault-tolerant/reliable networks and can show different approaches to the network designing. The works described there also show possibilities of the network design from different points of views.

The fourth chapter deals with the network topology design. There is an explanation of the design of the network with the different levels of the fault-tolerance in the different parts of the network. This algorithm is used as a basic part of the all modifications of the network design that are depicted in this chapter. The design of the logical topology is also described in this section since knowledge of the logical topology allows estimation of the end-to-end delay of data-frames. The modifications of the algorithm that allows to use a priory knowledge for the design of the network topology are shown in this chapter as well. Moreover, modifications that allow to use different physical layers for the network and work with the nodes with the different number of communication ports are also depicted in this chapter. The algorithm that is able to design the network with the same level of the fault-tolerance is described in this section as well (this algorithm was developed since algorithms mentioned in the literature are not able to ensure this fault-tolerance (see chapter: 2 **Reliability**). The design of the tree topology is important since the tree topology is often used in the telecommunication, computers or control technology. Therefore, this method is described in this section as well.

It is often necessary to change the existing network topology. Therefore, algorithms that are able to change the existing topology are described in the fourth section. These algorithms are able to preserve abilities of the original network or improve them if it is requested. They are able to design changes not only of the number of communication links but they are also able to design changes of the network topology if the whole groups of nodes are added or removed; this is a difference in comparison with the existing algorithm.

There are numerical or simulation results for every single described algorithm in the section fourth. There is also recommended setting of the algorithm for the physical topology design described in this section. These settings were gained thanks to the parallelization of the algorithm that allows to do enormous number of tests for the different numbers of nodes of the network. Even if the changes of settings were limited to the number of chromosomes, it was necessary approximately six months of computer time of six computers in average for gaining of these results. The way of algorithm parallelization is also described in the fourth section. There is also description of all genetic operators, fitness function and repairing algorithms that are necessary to use in the algorithms described in the fourth chapter.

The fifth section concludes the thesis and describes the contribution of this thesis. There is also a chapter that is dedicated to the possibility of the future research.

In the appendix is a description of the settings of the algorithms that were used for getting results described in this thesis. There is also an example of the results that are possible to gain if a crossover operator, which is described in the paper [1] mentioned in the section: 3 **Related works**, is used. This operator can produce a network that has not demanded level of the fault-tolerance, therefore it is depicted in the appendix as an example of the genetic operators that should be avoided. In the appendix, there is also possible to see all results of the design of the network that is used during the description of the simulation results. There is also possible to see the influence of the number of chromosomes on the quality results in the figures in the appendix. These results were gained thanks to the parallelization of the algorithm for the network topology design. In these figures, there is possible to see the best results that are possible to gain with the existing algorithm for this kind of the network and the best possible solution that would be possible to gain by the proposed algorithm as well.

1.3 Genetic algorithm

Following paragraph was adopted from the [11], [12],[13] in which is possible to find more detailed description.

The first applications of the evolutionary computing were used in the 60's. These applications solved real technical issues as a design of airfoils. Other solutions were dedicated to the optimization and machine learning. Evolutionary biologist used the evolutionary computing as controlled experiments in their research.

All these applications were used in order to solve specific task without intention to design a general theory that is able to describe optimization process. All methods were tailored for the specific tasks in contrary to the genetic algorithm developed by Holland and described in [13]. Holland abstracted the mechanism from the biology as a gene, chromosomes and genetic operators as mutation, crossover and selection that allows bigger reproduction of better results than those with worse results. Probably Holland's biggest contribution is the theory of *"Schemas"* and the application of population of chromosomes on which above mentioned operators do their job. The theory of "Schemas" tried to explain how the genetic algorithm works on the basis of probabilities. This theory allows explaining of the behaviour of the genetic algorithm and its operations. It is possible to understand Simple Genetic Algorithm [12] as a basic algorithm (also called Standard Genetic Algorithm). The SGA works as follows:

1. Create an initial population.
2. Calculate the fitness of the initial population.
3. Create new chromosomes with the help of genetic operators: mutation and crossover.
4. Calculate the fitness of newly created chromosomes.
5. Select chromosomes for the next generation.
6. If stop criterion was not met, continue to step No.3., if stop criterion was met stop the algorithm and return result.

The structure of the simple genetic algorithm is used for the algorithm or the network topology design that is described in this thesis.

Creation of the initial population For the run of the genetic algorithm, it is necessary to have an initial population. The initial population should cover the whole search space. Size of the initial population should correspond to the size of the search space in order to allow finding a good solution in reasonable time. There are two possible ways of the generation of the initial population. The population can be created randomly. This method is suitable if the structure of the good solution is not known or we are not able to create a structure similar to the good solution. In this situation, the whole initial population is created at random. Another method is suitable to use if we know the structure of possible solution and it is possible to create it. In this situation, the initial population is created at random but in order to looks like the structure of the possible solution. The example of this situation is a design of the ring topology. Every node in the ring topology must have degree $\deg(v) = 2$. In addition, the network must be connected. Then the all chromosomes are created randomly but they must meet the above described condition. This allows finding the solution faster than if the general randomly generated solution is used.

Evaluation of the initial population The initial population must be evaluated in order to find the quality of the initial population. This evaluation is later used for the selection of the chromosomes to the next generation.

Mutation The mutation operator is used in order to preserve diversity in the population and shift the solution in the search space. There is used the one bit mutation in the algorithm for the network topology design. The mutation flips the gene of the chromosome to another value. Every gene in the chromosome can be mutated with certain probability. The probability of mutation is usually very small [11], [12].

Crossover operation Crossover operator combines two chromosomes and tries to find better solution with the help of information stored in parent chromosomes. The one point crossover is used in the algorithm for the design of the network topology. Every chromosome from the population can be chosen and every border between genes can be used as a point of crossover.

Evaluation of the population The newly created chromosomes must be evaluated in order to gain the evaluation of the whole population. The evaluation of chromosomes allows to decide which chromosome is better than others.

Selection The selection chooses the chromosomes for the next generation of the genetic algorithm. The tournament selection [12] is used in the algorithm for the network topology design. Two chromosomes from the actual generation are compared and the better one is selected for the next generation. Moreover, the elitism mechanism is used. The elitism mechanism means that the best chromosome is always selected for the next generation even if the best chromosome was not compared with any other chromosomes. The chromosomes in the tournament selection are chosen at random for the comparison. Thus, it is possible that the best chromosome is not compared with any other chromosome and will not be chosen for the next generation. Therefore, the elitism mechanism is used in order to prevent losses of the best chromosome.

Stop criterion The stop criterion stops the whole genetic algorithm. In the algorithm for the design of the network topology, the number of iterations is used as the stop criterion. If the number of iterations reaches the value that is used as a stop criterion, the algorithm is stopped and the chromosome with the best evaluation is considered as the solution of the topology design. Another possibility is to use a stop criterion which decides according to the speed of evolution of the population e.g. If the population (best chromosomes etc.) has not been changed during the last ten iterations, stop the algorithm.

The genetic algorithm allows to find the near optimal solution in reasonable time even if the search space contains several local optimum and the search space has enormous size. Therefore, the genetic algorithm was chosen as the basic for the algorithms that are described in this thesis and allow to design the network topology.

Chapter 2

Reliability

Distributed control systems are more and more applied in the industry and have a great importance to an economical operation of the industrial technologies. A distribution of the control system does not only allow to control vast technologies in a huge plants but it can be also only possible solution for a gaining of the control systems with big enough performance. It is possible to imagine a control of a petrochemical plant for a petrol distillation, car production line, metallurgical plant or an offshore installation. The distribution of the control systems and control algorithms is natural in all these cases and in many others in which the controlled technologies are geographically vast and their control must be safe and reliable. Other situation, when the distribution of the control systems and algorithm is beneficial, is its application at systems that needs very fast response because of the safety or the technological reasons. Moreover, the distribution of the control systems brings cost savings thanks to the savings of infrastructure of the whole control systems. It is possible to use only one communication cable that connects parts of the distributed control systems among each other instead of using of enormous lengths of cables for an interconnection of sensors and central control system. Another possibility is to use remote inputs/outputs and connect them via communication cable to the central control system or use a combination of the mentioned possibilities. The distribution of the control systems is possible only thanks to a communication among parts of the system. It is necessary to share data among parts of the control systems for correctly working systems and therefore the communication has the crucial importance for the possibility of the control system distribution. The data sharing allows optimization of the technology control and better diagnostic of the controlled technology as well as the control system and thanks to that it is able to bring cost savings.

Although the distribution of the control system can bring a lot of advantages as a faster response or an easier maintenance of the system etc. it can also bring some disadvantages that are caused by the distribution of the control system. The control system could share data that are not only important for the optimal run of the controlled technology but they can be also important for safe operation of the technology (it could be a pressure in the tank, a concentration of methane in the air, temperature etc.). If data important for optimal run of the technology are lost, it can cause financial losses but if the safety data are lost it can cause not only financial losses but also it can literally mean a biological hazard or a threat to human lives. Therefore, it is necessary to ensure that the data exchange is reliable and data are uncorrupted. These demands are fulfilled thanks to the deterministic data exchange via communication protocols *(Profibus®, Modbus®, EtherCat®, Powerlink®)*.

Behaviour of the communication protocols is deterministically described by their standards and these standards also describe procedures that assure data integrity or prevent data corruption *(Cyclic Redundancy Check, SSH, etc.)*. In case that it is not possible to assure data integrity, these procedures should be able to inform that data are not correct. All these procedures do a great work with data exchange but they are not able to do anything if data are lost. The data losses can be caused by the two main reasons:

- Communication network is overloaded
- Communication network is interrupted

Basically, it is possible to say: the communication network is unreliable if data are lost and very probably the network is not designed correctly. It is not easy to prevent data losses when it happens. If the network is overloaded, it means that capacity of the network is too small, one must increase the network capacity or decrease data amount exchanged in the network. In most cases it is possible to use only the first solution, because it may be possible to add some communication link at the appropriate place but often it is not possible to reduce amount of exchanged data without a change of a communication technique. One can expect that data, which are exchanged, are necessary for a safe and reliable technology control. Hence, it is not possible to omit a part of this data and therefore they must be comprised in the data exchange or a method of the data exchange must be changed. Both these possibilities mostly need the change of a communication technique and the change of the communication infrastructure connected with the communication technique. However, the necessary change in the communication infrastructure may not be possible when the network is already built.

Data losses can be also caused by an interruption of the network or by a malfunction of the part of the control system. Since we expect the reliable control system, in which every part is important for the technology control, we can omit a malfunction of a part of the control system in our expectation. If some part of the control system does not work correctly then it is not possible to control the technology anyway and in that moment the communication among nodes is not so important for the correctly working system. Therefore, the fully operative control system can be expected and a malfunction of the control system can be omitted. Then it is possible to prevent data losses with the help of an addition of some communication link to the network.

The issue of data losses can be solved by an addition of the communication links into the network. Unfortunately, the solution of this issue is not as straightforward as it may seem. Hardware of the control system has a limited number of communication ports that can be used in the network, the technology used for the communication network has a constrained bandwidth and the addition of the communication link may cause data bottleneck at another place. Thus, an improvement of an existing communication network is limited and the behaviour of the improved network can be hardly better than the behaviour of the network that is properly designed according to the demanded attributes from very start of the design.

A design of the communication network for an industrial application is a quite complex task because inexpensive highly reliable networks are requested in the industry and customers have demands that are contrary to each other by nature (acquisition costs are one of the important attributes, another is a network fault-tolerance. If there is a small number of communication links, the acquisition costs are small as well as the fault-tolerance. It is possible to increase the reliability thanks to the increasing of the number of communication links then the acquisition costs are increasing as well).

One understands as a reliable network such a network that allows a communication among parts of the communication network without data-losses. Furthermore, the network should allow the communication also, if some communication link is interrupted, it means that there must be a redundant communication path for the interrupted one. The network for the industrial application can control technologies with very fast dynamic. Thus, data must be exchanged not only fast but also must arrive to their destination in specified time. The demand for data arrival in the specified time is the most important attribute of the hard real-time communication protocols as *Profinet®, EtherCat®, PowerLink®* for non real-time protocols it is only necessary to ensure that data arrive up to some specified timeout. This attributes of the communication network are directly connected to the capacity of the network and the capacity of every communication link. Moreover, every control system has a limited number of communication ports or can use different kinds of communication infrastructure than other parts of the control system. Therefore, it is not possible to place a communication link at some random position in the network but it is necessary carefully choose the appropriate way to connect part of the control systems among each other. We can see that it is not simple to design the communication network for the control engineering. One can design the network by himself for very limited number of devices. On the other hand, for bigger number of devices it is necessary to use some methodology that helps to balance antagonistic demands as reliability/fault-tolerance and acquisition costs.

2.1 Requested network

The network that should be applied in control engineering must meet several requirements. The network must be reliable from engineer's point of view. One can understand as a reliable network such a network that is tolerant to some communication link interruption. A tolerance to the link interruption (fault-tolerance) means that every node in the network is still able to communicate with the rest of the nodes even if some communication link is interrupted (It means that the network is still connected from the graph theory point of view [14]). Another request is a data-delivering in time (The network must have a big enough capacity for the data-transfer in order to avoid data losses and data delaying). The last important attribute is the cost of the whole network. The best possible solution of this optimization issue is a topology that is not only fault-tolerant and has a big enough communication capacity but it is also as inexpensive as it is possible. As we demand the application of the resultant topology in control engineering the main request for us is the tolerance to the link interruption and the big enough capacity of the network. Nevertheless, we do not want to design the topology that has the same level of the fault-tolerance in the whole network but we want the topology that has different levels of the fault-tolerance in the different parts of the network.

Since the design of the network topology belongs to NP-complete problem [3], [7] it is not possible to design the network in analytical way for medium and bigger networks. Therefore, it is necessary to use some heuristic algorithm as greedy [4], simulated annealing technique [1] or genetic algorithm [3], [6].

Heuristic algorithms allow fast finding of the near optimal network topology but they are not able to ensure finding of the optimal network topology. In general, it is not even possible to say that the solution found by the heuristic algorithm is close to the optimum. This is the main disadvantage of the heuristic algorithms; another one is that the algorithm must be designed according to the solved task. These disadvantages have an opposite in the main advantage of the heuristic technique and it is the speed of a solution finding.

There have been published many papers that focus on the telecommunication network topology design e.g. [8], [15]. These algorithms try to balance the reliability and costs of the network. Nevertheless, all these algorithms have not proposed solution that provides sufficient methodology for the network designer in the control engineering. Moreover, some algorithms are not able to ensure a design of the fault-tolerant network e.g. [1], [2] from the control engineering point of view and any of the algorithms is not able to design an inexpensive mesh network with the different levels of the tolerance to the link interruptions in the different parts of the network. This attribute leads to unnecessary expensive network. Thus, the new methodology for the network topology design is proposed in this work.

The designed network topology must satisfied two basic demands: the network must be reliable and allow the data delivering in time. Moreover, the network should

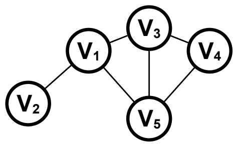

Figure 2.1: Mesh network with “core”

have as low acquisition costs as possible and can comprise nodes with the different and limited number of communication ports.

It is possible to understand to the reliability as a probability of the uninterrupted communication in the network. Nevertheless, it is necessary to know the reliability of each link for this purpose and it is possible only when the network is built. Therefore, the reliability, described by this definition, is hardly possible to use for new networks because we do not know the reliability of the communication links before the network is built. Thus, it can be used only for the verification of the reliability of the already built network. If we use it for the designed network before its actual construction it is only an estimation that may be correct but this estimation can be hardly used as a realistic reliability measure. Thus, the tolerance to the link interruption is used instead of the reliability.

The requested fault-tolerance is possible to set independently for every pair of nodes according to the controlled technology and control algorithm that runs in every part of the control system.

Part of the network, in which is the critical part of the control algorithm, needs more reliable communication among nodes and therefore bigger fault-tolerance to the link interruption (bigger number of independent paths) than other parts of the network. Node that archives data is not critical for controlled technology and needs one or two independent paths for the communication with the other parts of the network. Similar situation is for the communication between the network of the control system and office network that provides data for managers. This kind of communication is not needed to be as reliable as the communication in the control network. Thus, one communication path is enough reliable for this purpose. The network designer can set the demanded fault-tolerance for every pair of nodes thanks to the knowledge of the controlled technology and describe the demanded fault-tolerance in the matrix of fault-tolerance M.

It is possible to see an example in the Fig. 2.1. Node V_2 represents a station that allows a supervision of the controlled technology but not the actual control of the technology. It provides statistical data, which describe efficiency of the controlled technology, for managers of factory or supervisors of the technology via public access. Node V_1 archives data and helps to communicate to a “core” of control system that include nodes V_3 and V_5. In the node V_4 runs not so critical part of the control algorithm as in nodes V_3 ans V_5.

2.2 Reliability

The reliability of the network is connected with a possibility of uninterrupted communication even if some communication link is interrupted. If the network is disconnected then some node cannot communicate with another node in the network and therefore the data-exchange is not possible. It is possible to omit the reliability of the nodes because the network is used in control engineering and the main goal is not only communicate among nodes but allowing of the technology control. The network is only tool that should allow an exchange of data that are necessary for the control of the systems. If node is destroyed, it cannot control the technology and therefore in this case the communication among nodes is not important any more (even if the network is fully operative there are any data for the data exchange if the nodes do not work). Therefore, the fully operative nodes are expected.

2.2.1 Ring topology

The basic and common way of the reliable network topology design is an application of the ring topology (see Fig. 2.2).

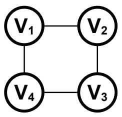

Figure 2.2: Ring topology

The ring topology allows uninterrupted communication even if a communication link is destroyed because there exist two independent paths for every pair of nodes. This topology can be used for the control systems that include small number of nodes.

In the Fig. 2.3 and Tab. 2.1 is possible to see that the reliability is dependent on the link reliability and the number of the communication links of the network (Number of links is the same as nodes in the network with the ring topology).

The Monte Carlo method was used for data gaining. It was done 10000 tests for every number of nodes of the network. Communication links fail statistically independently and the link reliability is the same for all of them. It is possible to see that the terminal reliability for the network with the ring topology, which comprises 10 communication links with reliability 0.9, is 0.74. This reliability can be sufficient for non critical parts of the network but for the part that is important for the safe control of the technology it is not enough. The huge disadvantage of this method is a necessity of the link reliability estimation. We can count the link reliability if the network is already built and we have statistical data from the real network operation. If we do not have the real data, we can try to estimate them according to other similar network.

$LinkReliability$	0.8	0.85	0.9	0.95
$Nodes$	Network Reliability			
5	0.74	0.84	0.91	0.98
6	0.66	0.78	0.88	0.97
7	0.58	0.71	0.85	0.96
8	0.51	0.66	0.82	0.94
9	0.44	0.60	0.78	0.93
10	0.38	0.55	0.74	0.91
11	0.33	0.50	0.69	0.89
12	0.27	0.45	0.65	0.88
13	0.24	0.40	0.62	0.86
14	0.19	0.36	0.58	0.84
15	0.17	0.31	0.54	0.83
16	0.14	0.28	0.52	0.81
17	0.12	0.25	0.48	0.79
18	0.09	0.22	0.44	0.78
19	0.08	0.20	0.41	0.75

Table 2.1: Reliability - Ring topology

The estimation of the link reliability can be very inaccurate since the networks can be applied in different environment. Thus, it is necessary to use some other method.

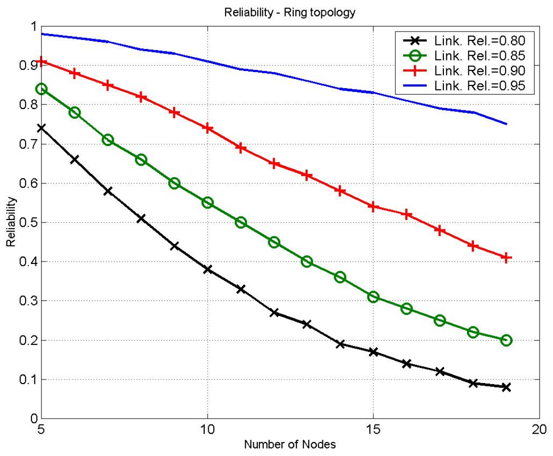

Figure 2.3: Reliability - Ring topology

One method was introduced in [16]. There was used a novel approach for the network topology design. The cross-entropy method [17] was used for the calculation of the network reliability in [16]. It is very useful method that allows to count a probability of rare events as a communication link interruption. Method described in [16] expects minimally bi-connected network and solves the issue of the network topology design with the unreliable communication links (every communication link has its own reliability. In proposed case, every link has the same reliability as the others). A disadvantage of the proposed method is necessity of the link reliability estimation. The algorithm described in [16] is not able to design a network with the different levels of the fault-tolerance in the different parts of the network or design a mesh network that could include a network part with the bus topology as is in the Fig. 2.4.

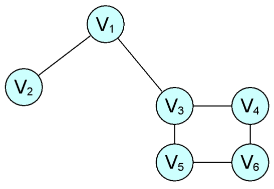

Figure 2.4: Mesh network

Another solution of the network topology design problem was suggested in [8]. The concept of the *k-terminal reliability* was introduced in [8]. The novel method of the reliability calculation was described in [8].

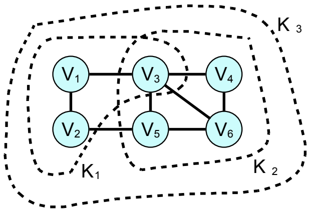

Figure 2.5: K-reliability

This method allows to design the network with the different reliability in the different parts of the network but it is not able to ensure the different fault-tolerance in the different parts of the network. The method expects that the network is split into k parts (set of the nodes - see Fig. 2.5); for every set of the nodes is possible to have different demands for terminal reliability (probability that all nodes in the set are able to communicate with all other nodes in the set). If the reliability of the communication links is known, then it is possible to count the reliability of every set of nodes with the help of the Monte Carlo method.

The Monte Carlo method is used for the terminal reliability calculation of the network designed by the genetic algorithm. The Monte Carlo algorithm generates set of interrupted communication links of the designed network and the node interconnection is verified for every set of k . Then the algorithm is able to calculate the *k-terminal reliability* for every $k - set$ thanks to the providing of numerous tests. This method again expects the knowledge of the link reliability and it is not able to design a network with the different levels of the fault-tolerance in the different parts of the network or a mesh network that could include a network part with the bus topology as is in the Fig. 2.4 neither as method described in [16]. Algorithm is not able to ensure the fault-tolerance because of the nature attributes of the Monte Carlo method. Monte Carlo method is based on the application of the numerous statistical independent tests. Result of these tests is able to describe tested hypothesis with confidence interval that is dependent on the number of the statistical independent tests. Naturally, one can see that the Monte Carlo method is not able to ensure 100% network reliability if certain number of communication links fail and therefore it is not able to ensure that there is not such a configuration of the interrupted communication links that is able to disconnect the network. Therefore, methods based on the Monte Carlo algorithm are not able to ensure the network fault-tolerance.

Methods that are able to ensure fault-tolerance are based on the graph theory. One was introduced in [1]. There was used a degree of nodes as a fault-tolerance measure. Unfortunately, the proposed method is possible to use only at very constrained conditions.

- Node degree is used as fault-tolerance measure
- Node degree is used as fast approximation of another reliability measure

2.2.2 Node degree as an approximation

An application of the node degree as a reliability/fault-tolerance approximation has a great advantage: it is a speed of finding of the node degree. It is very fast and thanks to that, it allows a creation of very fast algorithms. An algorithm based on this methodology would be always faster than any other would be (other algorithm must find node interconnections firstly and then they work with the topology described by these connections). However, the test of the node degree is possible to use as a fast approximation but as the final test of the reliability is not as useful as it could seem to be.

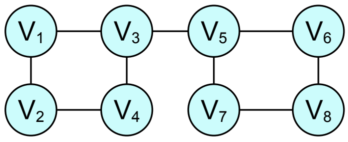

Figure 2.6: Mesh - non 1FT

In the Fig. 2.6 is a network that satisfies demands:

$$\deg(V_i) \geq 2 \quad \text{for} \quad V_i \in V \tag{2.1}$$

It is possible to use an equation (2.1) as a fast “approximation” of the minimal edge connectivity according to [1]. If the minimal edge connectivity is equal to 2, then the network is $1 - fault - tolerant$ and the equation (2.1) should ensure it according to [1]. However, the equation (2.1) is not possible to use as a real approximation of the minimal edge connectivity. One can see that it is possible to disconnect network if only one link is interrupted (link between V_4 and V_5). Then the network depicted in the Fig. 2.6 does not have the minimal edge connectivity equal to two but to one and therefore the network is definitely not one fault-tolerant and the equation (2.1) is possible use only as a fast rough approximation of the edge connectivity and the fault-tolerance as well.

2.2.3 Node degree as a reliability measure

An equation (2.1) was used in [1] as a fault-tolerance measure. Moreover, there was proposed a heuristic that repairs unreliable topologies (those with $\deg(V_i) < 2$). The equation (2.1) should ensure that the designed network should be $1-fault-tolerant$ (a failed link does not interrupt the communication among nodes). Nevertheless, it is possible to see in Fig. 2.6 that the equation (2.1) is not able to verify whether the network is really fault-tolerant. The network in the Fig. 2.6 should be $1 - fault - tolerant$, but it is not true because an interruption of the link between V_1 and V_2 disconnects a network into two parts that are not able to communicate with each other anymore. Thus, the equation (2.1) is possible to use only as a necessary condition for the fast decision whether the network may be $1 - fault - tolerant$ but it is not possible to use it as a sufficient condition for the verification whether the network is really $1 - fault - tolerant$. It is necessary to use different condition for verification whether the network is $1 - fault - tolerant$.

$$\deg(V_i) = 2 \quad \text{for} \quad V_i \in V \tag{2.2}$$

(2.2) is the same condition as for the minimal Euler graph [14] and it really ensures that the network, which corresponds to the (2.2), is $1 - fault - tolerant$. The equation (2.1) is possible to use as a fault-tolerance measure if there are also other conditions (it cannot be use as a test of network that has similar structure as the network in the Fig. 2.6).

We can see that an application of the node degree as a fault-tolerance measure is very limited and cannot be used for the complex demands such as different fault-tolerance in the different parts of the network. The huge advantage of this method is that it is not necessary to estimate the communication link reliability and it is possible to use only knowledge about the structure of the network topology. (2.2) and similar equations allow to find only the network with the same level of the fault-tolerance but it does not allow to verify whether the network has different fault-tolerance in the different parts of the network. The verification of the different fault-tolerance in the different parts of the network is described in the thesis.

Chapter 3

Related works

The first papers concerned with the network topology design were published at early 70's. Some of them are about the design of the topology of pipelines [18] but most of them dealt with the possible design of the ARPANET.

Gerla and Kleinrock [19] introduced several algorithms for the network topology design (Concave-Branch elimination), flow assignment and routing policy (Flow-Deviation algorithm). The Flow Deviation method is a hill climbing method that expects continues cost function. In case that there are more local extremes, it is not possible to ensure the global extreme finding. If the cost function is not continues, one must find an interpolation of the cost function. The proposed algorithm for the topology design (Concave-Branch elimination) iteratively uses the flow-deviation algorithm for the finding of several local extremes. The proposed method is able to design the reliable network that is tolerant to the interruption of the link. Thus, the method is possible to use for small networks with the small probability of the link interruption. The method is not able to ensure the design of the network topology with a bigger fault-tolerance than one or different fault-tolerance in the different parts of the network. The algorithm cannot integrate a priori knowledge of unsuitable topologies as well. Hence, it is possible that the algorithm designs the network with the topology that is unsatisfying for us for some reason. The most important contribution of this paper is an interpolation of the end-to-end delay of the traffic flow of the packet switched communication. Model of this interpolation was gained according to the data from ARPANET network. This interpolation of the M/M/1 system (according to Kendal's notation) is still used for the calculation of the average time delay of systems that has Poison's distribution of interarivel time and service time and use one communication link.

Dutta and Mitra proposed in [20] an algorithm that is able to design the network with the different fault-tolerance in the different parts of the network and optimize the acquisition costs with constrains to the fault-tolerance. The algorithm is a hybrid of mathematical programming (Mixed Integer Programming) and heuristic algorithms.

The mathematical programming generates initial solution of the topology that is later changed by the iterative algorithms that ensure fulfilling of the connectivity requests, degree constraints and assign the flows to the communication paths. The whole algorithm works as an iterative task that allows running the algorithm again if the demands are not met. The algorithms contain quite complex corrective algorithm that prevents heuristic algorithms to cycle the same solution among heuristic algorithms (e.g. the connectivity task proposes the solution that does not meet the degree constrain. After that, the degree task repairs the network to meet the degree limitations; then connectivity task finds out that the connectivity demands is not fulfilled and proposes the same solution as before. This could lead to infinite iterations without the corrective algorithms). The algorithm is able to design the topology in such way that the average delay of all packets should be under demanded deadline. Unfortunately, the algorithm works with the average delay of all packets and it is not able to ensure the delay for every single packet. It means that it is possible that some packet is delivered in four times longer time than it is demanded and another is delivered in quarter of the demanded time. In this situation, which is not definitely corresponding to the demanded behaviour, the average delay will correspond to the maximal permitted delay and we will not have any sign about impropriety of the designed network. Then, the error in design appears after actual construction of the network and very probably in the situation in which the communication fails. In case, that the network is used in the control engineering, this error can cause the serious issue or life threats. The algorithm is also not able to work with the different physical layers of the network. The proposed algorithm [20] is also not able to design an expansion or reduction of the existing network.

Dengiz at all [21] proposed an algorithm for the reliable network design. The algorithm uses a genetic algorithm for purpose of the network design. The comparison of the algorithm with the Branch&Bound method was made at 79 testing sets. These tests proved that the genetic algorithm outperformed the Branch&Bound method. The algorithm focuses only on the design of the topology that should be reliable. The algorithm uses a fast estimation of the reliability that speeds up the reliability calculation (the reliability is exactly evaluated only for the most promising chromosomes). Unfortunately, the algorithm is not able to design $1 - fault - tolerant$ (1FT) network even it seems to. Algorithm uses the degree of nodes as an interpolation of 2-edge connectivity and consequently as a measure of the fault-tolerance. A condition $\deg(V_i) \geq 2$ is used for this purpose. In fact, this condition is not able to verify $1 - fault - tolerance$. The most of networks that meet the condition are really $1 - fault - tolerant$ but there exist network topologies that fulfill the condition but they are not $1 - fault - tolerant$ (see chapter 2 **Reliability**).

Dengiz at all [21] proposed a corrective algorithm that should repair networks that are not $1 - fault - tolerant$ (according to the condition $\deg(V_i) \geq 2$) and were made by the genetic operators (Initial population meets this condition). This correcting algorithm ensures that all chromosomes, which are in the chromosomes set, are $1 - fault - tolerant$ (according to $\deg(V_i) \geq 2$). Moreover, the algorithm does not solve issue of the constrained number of the communication ports or the delay of the delivered data. Therefore, the results gained by this algorithm can be used only as candidates for a future investigation of its attributes.

Ko at all [6] proposed a method for interconnection of ten Chinese cities via fibre cable. The method is based on application of the genetic algorithms for solving of several issues. The method solves issue of the fault-tolerant interconnection of cities; minimal two independent communication paths are demanded for every pair of cities. The solving of the average time delay, flow assignment and link capacity issue is integrated into the algorithm as an optimization subroutine in this algorithm. The advantage of this algorithm is that the optimizing subroutines are integrated directly into the main task of the network topology design. It means that if the network does not accommodate demands for the in time data delivery, the design algorithm can try to change the topology immediately in such way that offspring of the current insufficient network meet demands for in time data delivery. On the other hand, this direct influence is also disadvantage in the different point of view. The verification of the in time data delivery is very time demanding since it uses a genetic algorithm as well. The verification must be done for every newly created chromosome in every iteration of the main algorithm and it needs a lot of time. Algorithm outperformed the branch exchange method [6] with which the comparison was done. Results of the branch exchange method need more fibre cables. Regardless to the bigger need of the fibre cable, the average delay of data was bigger in comparison with the proposed genetic algorithm [6].

The proposed algorithm can be very useful for the design of the backbone of the computer network but its application in control engineering could be very constrained. The algorithm is able to ensure only $1 - fault - tolerance$, but it is not able to ensure the design of the network with the different fault-tolerance levels in the different parts of the network (it is not able to ensure the design of the network with bigger or smaller level of the fault-tolerance than one fault-tolerance). The algorithm also does not incorporate constrains of the number of the communication ports. The main reason why this algorithm is unsuitable for the application in control engineering is the verification of the end-to-end delay. The algorithm verifies only the average time delay in the whole network and it is not able to ensure the maximal delay for every data-flow in the network. This attribute directly disqualifies the algorithm for an application in control engineering especially in the real-time systems. The average value of the delay does not say anything about the real behaviour of the network under demanded load (e.g. a half of the delays can be two time bigger than permitted value and half can be two times smaller than the maximal permitted delay and according to the delay evaluation in this algorithm is everything all right). Therefore, this algorithm is not possible to use for the network design in the control engineering. The designed network could be inexpensive, one-fault-tolerant with the satisfying average time delay but still not suitable for the application in control engineering since the delay of some data-flow could exceed the maximal permitted value.

Jan at all [22] suggested an algorithm for the design of the reliable computer networks. This algorithm ensures the global optimum finding in contrary to other algorithms (those based on heuristics, genetic algorithm and so on). The method is based on the Branch&Bound algorithm that is also the main disadvantage. The Branch&Bound algorithm searches through the whole search space. This search is very time demanding and needs a huge amount of the memory for an expansion of a tree of the possible solutions. Jan at all [22] proposed a heuristic algorithm that allows to speed up the whole algorithm. This heuristic uses the basic attribute of the network: the network must be connected for the communication among all nodes and the essential condition is that the network must contain minimally $N-1$ communication links for N nodes. The upper bound of the possible network reliability is used as another heuristic rules for the algorithm speeding up (where the upper bound of the possible reliability depends on the node degree). Then, the upper bound of possible reliability must be bigger than demanded network reliability [22]. The exact reliability of the designed network is calculated only if the constraints for the number of communication links and upper bound of the reliability are satisfied. These heuristics rules allow significant reduction of the search space and thanks to that also the time that is necessary for the network design. Nevertheless, the algorithm is not able to design the network with the different reliability in the different parts of the network or even to ensure the network fault-tolerance. The algorithm also does not incorporate the degree constraints in spite of the application of the node degree for the calculation of the upper bound of the network reliability. Even if the heuristic rules are applied, the algorithm is hardly able to solve the design of bigger networks, because of enormous growth of the search space and the combinatorial tree of the Branch&Bound method as well. The algorithm does not solve the issue of data delivering time at all. It just design the network that meets demands for the network overall reliability without a consideration of the data delay issue. Therefore, the algorithm is not suitable for the application in control engineering as well as previous cited algorithms.

The algorithm for the expansion of the existing network was proposed by Kumar at all [23]. The method uses a genetic algorithm as a core for the optimization of the network expansion problem. The network expansion problem arises very often in control engineering as well as in the computer networks. This necessity often occurs after some changes of the controlled technology. These changes often lead to the changes of the control system such as a reduction or an expansion of the control system or number of parts of the control system. The genetic algorithm described in the paper does not use very efficient chromosome coding because the coding allows to create self-loop connection of the node. This kind of coding causes unnecessary chromosome length and therefore wasting of the computer memory. The application of the common genetic operators can create unfeasible chromosomes (e.g. node is connected to another node according to a gene of chromosome but according to another gene the nodes are not connected). Therefore, a correcting algorithm was also proposed. This correction repairs unfeasible chromosomes and allows to design feasible chromosomes. The algorithm is able to expand the network under reliability and nodes degree constrains even if it is not able to design the network with the different levels of the reliability in the different parts of the network. Unfortunately, the algorithm is not able to ensure the design of the fault-tolerant network as well as it is not able to use a priori information about the inappropriate topologies. The algorithm does not also solve the issue of data-flows and delay of the data delivering. Therefore, this algorithm is not possible to use in control engineering.

The algorithm for the design of the LAN networks was proposed in [7]. The algorithm solves an important part of the computer network design such as a computer assignment to clusters in order to minimize the network load among those clusters. The algorithm uses an original coding in which chromosome represents a virtual Huffman tree. This coding is longer than the minimal possible coding but has a great advantage in simplicity of finding of a path between segments of the network described by the Huffman tree. The algorithm is based on the genetic algorithm. The genetic operators change the chromosomes and they can destroy the virtual Huffman tree. Therefore, the algorithm also implements a correcting procedure that changes an invalid virtual Huffman tree into the correct form of the virtual Huffman tree.

The algorithm [7] tries to minimize an amount of the data transferred among clusters of the network by the optimal placing of computers in the clusters. The average time delay for the network traffic is employed as an optimality measure. The fault-tolerance and the delay for every data-flow is not solved in this algorithm because of demands for clustering algorithms (there is no expectation of the independent communication paths among clusters, and there should be minimal traffic load among clusters if the computers are correctly placed into the clusters). This is possible to assume for the common computer office network, in which common behaviour is that most of the traffic is among a few computers [7]. This is an assumption that is not valid for the network in the control engineering since there is an expectation that parts of the control system communicate with the other parts of the network. Then, the assumption of possible minimizing of the network delay by the optimal distribution of the parts of the control system into the clusters is not possible to use.

Saha and Chakraborty [24] developed an algorithm for the computers network enhancement. This method uses a genetic algorithm with the common genetic operators for the purpose of the network enhancement under constrained costs. This approach allows enlarging network with as low costs as possible and a reduction of costs for the network enhancement. Proposed algorithm showed a good ability of the network enhancement but unfortunately it is neither able to ensure a design of the fault-tolerant network nor reliable network. The algorithm works only with the budget as an optimized variable but does not incorporate other demands. It means that an application of the extended network can cause serious issues. The algorithm does not incorporate possible physical limitations of the network caused by the constrained node degrees. Then the designed network may not be build since the nodes do not have enough possible communications ports available. Another very important attribute is not considered as well. The traffic delay should be one of the most important quality measures for every network. Only the traffic delay is able to say what happens in the network whether there is some bottleneck for the network traffic that may cause data losses and consequently a malfunction of the whole system that uses a network for the communication. Since the algorithm does not use any calculation or even an estimation of the delay, the proposed algorithm is not possible to use in the control engineering. The proposed algorithm should not be used for the design of any network at which we expect reliable data exchange because algorithm does not allow to find the network behaviour before the actual construction of the network.

An algorithm for the design of the fault-tolerant network is described in [1]. The algorithm uses a common genetic algorithm [12] that incorporates repairing heuristics that should ensures that all chromosomes in the set describe feasible solutions. The heuristic is applied at chromosomes that are results of genetic operators (crossover and mutation). Thanks to that, the algorithm should design the fault-tolerant network that allows uninterrupted communication among nodes and allows in time data delivery. The algorithm seems to ensure $1 - fault - tolerance$ in the whole network and use edge connectivity as a fault-tolerant measure. Unfortunately, it does not use the edge connectivity directly but uses an approximation instead of the edge connectivity and therefore the algorithm does not ensure design of the $1 - fault - tolerant$ network since the network can be disconnected by an interruption of just only one communication link (see 2 **Reliability**). The repairing heuristic rules are not able to repair all non fault-tolerant networks. Moreover, the repairing heuristic can produce non fault-tolerant network and does not inform about that **(see Appendix)**.

The algorithm [1] designs only the physical topology and does not verify whether the network allows the data delivery in time, hence the proposed network may not allow in time data delivering and cause a malfunction of the controlled system. Therefore, the algorithm is not very appropriate for the application in the control engineering area. Furthermore, the algorithm also does not use the node degree as a limitation of the network design and expects the infinite number of communication links at every node.

In the paper [1], a comparison of different methods of the network design has been done. An approach based on the genetic algorithm was briefly compared with Branch&Bound method and simulated annealing technique. There has been pointed out that the Branch&Bound method finds the global optimum but it is very time exhausting search technique. The simulated annealing is faster than the Branch&Bound method but often finds only a local optimum. On the other hand, methods based on the genetic algorithm are very fast but are not able to ensure the global extreme finding. In opposite to simulated annealing method, the genetic algorithm provides parallel search and therefore the global extreme finding is more probable than for the simulated annealing technique.

It is possible to use different kinds of communication technique in the modern communication systems. It could be peer to peer communication, when the messages are sent directly from the source node to the destination node. This kind of communication has an advantage if the most of the communication is between two nodes and every node is the receiver of different data from the source nodes. This kind of communication has a great disadvantage if the source node sends the same data to different receivers. At that moment, the data transfer needs a lot of time in comparison with a method at which data are sent as a multicast message to all receivers. If it is necessary to send the same data to different receivers, the multicast messages easily outperform the peer-to-peer communication. Therefore, the multicast method of the communication should be considered during the network topology design. An approach for the design of the logical topology for the multicast messages is proposed in [10]. There were described two different methods for a finding of the multicast tree. First of them applies the Branch&Bound algorithm for a finding of the delay bounded multicast tree. The algorithm incorporates extended Dijkstra's algorithm [14] for the searching of possible solutions. The whole algorithm was named Dboc. Dboc is able to find the optimal solution of the multicast tree design. Unfortunately, it needs exponential time for finding of a solution in the worst-case scenario [10]. It means that for bigger networks it may not be able to find a solution in reasonable time because of the size of the search space. Therefore, another algorithm is proposed in [10] as well. It is based on the genetic algorithm. The genetic algorithm ensures a fast solution finding in contrary to the Branch&Bound algorithm but the genetic algorithm is not able to ensure that the found solution is the optimal one. The genetic algorithm incorporates the rules that ensure disconnecting of all communication cycles in the network since the expected result is the multicast tree. The steady state of the results of the genetic algorithm was used as a stop criterion of the genetic algorithm. Hence, the genetic algorithm is able to provide the solution even if the solution is not the optimal one.

Since the algorithm [10] expects the application of the common nodes without ability of a usage of the communication tree backup, the algorithm does not solve issue of the reliability or the fault-tolerance. Algorithms apply the average delay as an evaluative function. This is the big disadvantage for the application in the control engineering since the average delay does not say anything about the maximal delay in the multicast tree and therefore algorithms are not able to ensure data delivering until the maximal permitted time elapses. It means that the algorithm is not suitable for an application in control engineering.

In the network at which it is not possible to expect every node with the switch ability it is necessary to place switches at places that allow to use as small number of switches as possible and decrease the price of the network in this way. A method for the switch placement was proposed in [25]. Two methods were described for the finding of the optimal switch placement. The first one applies heuristic rules for switches placement (it creates list of the nearest neighbours for every nodes and then place randomly as less switches as possible, check the network connectivity, find shortest path for every data-flow and evaluate price of the network, if the price is smaller than the best one reached, store network. This process is continually repeated until the number of iterations is exceeded). Another proposed algorithm used the genetic algorithm for the switch placement problem. The placement of the switches is chosen randomly and genetic algorithm has not any special rules for the switch placement in the initial population.

According to the published results, the genetic algorithm outperforms the heuristic one but both of them have the same disadvantages. The algorithms do not solve the network fault-tolerance or even the reliability issue. The independent communication paths are created at random since placement of switches is done randomly as well. It means that the algorithm is not able to ensure the certain level of the reliability of the network. Moreover, algorithms do not solve issue of the data-flow delay. Therefore, the algorithms are not appropriate for the application in control engineering since they do not provide any information about possible data delays in the network and are not able to ensure any maximal delay in the network. Then the network may have some bottleneck for data-traffic and data can be delivered too late and control algorithm can fail because of the late data delivery.

A finding of the network reliability is time demanding but it is required for the solution of the network topology design. The algorithm for the exact calculation of the network reliability was proposed in [26]. The algorithm decomposes the network to the tree that contains parts describing smaller part of the network. Then the algorithm creates list of all possible malfunctions of this tree and iteratively calculate the reliability of the uninterrupted communication. In case, that the algorithm is not able to find a whole solution, it estimates lower bound of the reliability. There is 5% uncertainty of lower bound estimation according to the numerical results [26].

The method described in [26] calculates the reliability of the network in which the link reliability is known. This attribute is exactly known only at already built networks after certain operational time. On the other hand, the link reliability can be only estimated in the networks that are planned to be built and may not be accurate enough. Therefore, the described methods give us proper information only for already built networks. Moreover, the algorithms count only probability of uninterrupted communication but does not consider whether the network allows in time data delivery. Therefore, it is possible that the network meets demands for the reliability but when the network is applied in real word situation, the network can be useless since data are not delivered into the destination in time or can be lost. Therefore, the application of this algorithm in area of the control engineering is not very appropriate.

The goal programming is one of algorithms that is very useful for the network topology design since it allows to set demanded attributes of the network. This method was used in [8]. The algorithm designs the network that is as inexpensive as possible while it meets demands for the different levels of the reliability in the different parts of the network. Described method uses a Monte Carlo algorithm for the reliability verification. It allows designing a network with the different levels of the reliability in the different parts of the network but it is not able to ensure the fault-tolerance see 2 **Reliability**. The application of the goal programming allows setting of the different priority of every reliability level. The algorithm uses common genetic operators that can create unfeasible solutions. Hence, every solution created by the genetic operators is checked whether it is feasible or not and only feasible solutions are used in the next generation. Unfortunately, the algorithm [8] expects no limitation at the node degree. It means that the algorithm may design the network in which should be more communication links at some node than it is possible. Moreover, the algorithm does not incorporate a verification of the traffic time delay. Therefore, it is not able to find whether the network allows in time data delivering. Thus, the algorithm is not appropriate for an application in control engineering.

The method for the fault-tolerant network topology design was described in [15]. The algorithm is based on the simple genetic algorithm [12] and applies common genetic operators. The algorithm incorporates an iterative repair heuristic procedure that modifies the unfeasible solution into the feasible one because of the common genetic operators can produce unfeasible solutions. Moreover, the repair heuristic algorithm is applied for every solution created by the genetic operators and modifies chromosomes to meet demands for the fault-tolerance if it is needed. The number of independent spanning trees is used as a measure of the fault-tolerance (if some communication link in the spanning tree is interrupted, then other spanning trees are uninterrupted and allow the communications among nodes). [15] pointed out that solutions of the network topology design brings an issue of the choice of the suitable evaluative function, especially if the penalty function is used. There was shown [15] that the wrong penalty function can cause better evaluation of the unsuitable solution than the suitable one. The algorithm applies technique of the reliability upper bound calculation for evaluation speeding up. Even if the speeding up technique is used, the algorithm needs 9000 seconds for the design of the network comprises 15 nodes (the 15 biggest German cities). The method is compared with a greedy algorithm for the network topology design that is outperformed by the genetic algorithm especially for the design of the reliable network of the 15 biggest German cities. The algorithm does not include a solution of the degree limitation issue. The algorithm solves the important task of the design of the reliable network topology. However, it does not incorporate another important attribute of the network such as the delay of the data-flows in the network. It means that after the search of the reliable network the algorithm may produce a network that does not allow in time data delivery. Therefore, the algorithm described in [15] is not possible to use in the control engineering since the delay of every data-flow has a big importance in the control engineering because the late data-delivery can cause malfunction of the control algorithm and consequently malfunction of the controlled system.

The algorithm that incorporates the fault-tolerance and the time delay issue was proposed in [27]. The algorithm is based on Bicriteria genetic algorithm (one criterion is for the network price and another is for the delay of the network traffic in the network). The algorithm finds a Pareto optimal set and offers a heuristic algorithm for the choosing of the appropriate solution among solutions of the Pareto optimal set. The algorithm does not include the node degree constraints and therefore expects that the number of communication links can increase to infinite at every node. This assumption is not naturally possible. The algorithm uses the average end-to-end delay as a measurement of the network behaviour under demanded load. The average delay describes general network behaviour but it is not able to say whether the network allows in time data delivery for every data-flow. It means that a data-flow may need two time bigger time than it is permitted and another one may need a half of the time but the average time delay still meets the demands. Furthermore, the average time delay for the configuration, in which some data-flow has a bigger delay than permitted, can be smaller than for the network in which all data-flows have smaller delay than it is permitted but the average one is bigger than in the first case. Then the network with a smaller delay is evaluated as better one even if it does not satisfy the demands for the maximal permitted delay for some data-flow. Thus, the algorithm is not very useful for control engineering because it is not able to ensure demanded delays in the network and it can cause the malfunction of the control system and consequently a malfunction of the controlled technology.

A general algorithm for the design of the fibre cable networks was proposed in [3]. The algorithm proposes a method for the topology network design and the network enhancement. The algorithm uses a goal programming for the design of the network with the demanded attributes as they are: reliability and minimal average delay of the data delivering. The algorithm is able to design the network with the different levels of the reliability in the different parts of the network. The Monte Carlo algorithm is applied for this purpose. The Monte Carlo algorithm is not able to verify the network fault-tolerance and especially not the different levels of the fault-tolerance in the different parts of the network. The algorithm verifies the delay of the data-frames in the network and thanks to that, it is possible to anticipate behaviour of the network before the actual creation of the network. Nevertheless, the algorithm uses the average time delay calculation for this purpose and it means that we know only the average behaviour of the network but we do not know the actual behaviour of the network under demanded load. There can be a data frame with very small delay and the other one that exceeds expected value but the average delay says that everything is all right and there is no issue. The real behaviour can be found after actual built of the network. The algorithm offers also the enhancement of the already existing network. The network enhancement applies the limitation for the average time delay of the data-frames and the budget for the network enhancement. However, the algorithm does not incorporate the node degree constrain and therefore it is possible that the network enhancement will not be possible to apply at the network since the communication nodes do not have enough communication ports available. The algorithm tries to design the network in which the node degree is close to the demanded node degree but there is no strict rule which says: “If the node degree is higher than the permitted degree then the network is inappropriate”. The algorithm does not allow designing an extension or a reduction of the number of nodes of the network or even use a priori information for the network enhancement. This can be a huge disadvantage for the enhancement procedure since people who demand the network enhancement can have knowledge about impossible solutions of their issue but the algorithm can design the network enhancement the same as the inappropriate one. Method proposed in [3] is not possible to use in control engineering since it does not ensure the network fault-tolerance and does not include the limitation of the node degree. It is not able to design the number of nodes addition/reduction. Algorithm is able to add or remove the number of communication links.

The algorithm that should design a fault-tolerant computer network was described in [2]. The algorithm uses a simple genetic algorithm [12]. The algorithm designs the network with as small costs as possible while the network should be fault-tolerant and have small average delay. Algorithm does not verify the delay of every data-frame therefore the algorithm is not able to ensure so that the designed network allows keeping delay of every data-frame under the maximal permitted value. Moreover, the algorithm uses the node degree as an approximation of the edge connectivity. This method has very constrained usage and the application described in [2] does not ensure the real fault-tolerance (see 2 **Reliability**). Therefore, the designed network may not be fault-tolerant and customer does not know it. This can cause a serious issue if the network is used for the delivering of the safety data. Not only that data can be delayed unexpectedly but the data may not be even delivered because the network can be interrupted by the only one link malfunction even if the network should be one fault-tolerant (see 2 **Reliability**). The method described in [2] should not be applied in control engineering since it is able to cause not only serious financial losses but also casualties if the network is applied for safety purposes.

A methodology for the network topology design based on the greedy algorithm was proposed in [4]. There are three greedy algorithms described for the design of the fault-tolerant network. All algorithms use the same cost function that is a sum of a function of the link capacity and a function of the link length. Evaluation of the network according to this function and this evaluation can cause that link with smaller capacity and small length will have better evaluation than the link with higher capacity and bigger length. Therefore, it is possible that the designed network will have communication links with small length as well as small capacities and therefore will not be able to handle with the demanded load since there are not any penalty functions for case if the link capacity is smaller than demanded. The quality evaluation via sum function is not able to ensure network with the big enough capacity. Proposed greedy algorithms verify the fault-tolerance with the help of the attributes of the graph theory. Author verify the number of independent spanning trees (if network is 1-fault-tolerant, there should be 2 independent spanning trees at least). This verification allows designing the fault-tolerant network but does not allow designing the network with the different levels of the fault-tolerance in the different parts of the network. Hence, the designed network will be unnecessary expensive if the network with the different levels of the fault-tolerance satisfy our demands. One can imagine that it is necessary to reach 3-fault-tolerance in the part of the network and in the rest of the network 1-fault-tolerance is satisfying. If the described approach is used, one must design the whole network as 3-fault-tolerant and therefore more expensive than would be necessary. The proposed algorithm is not suitable for the control engineering since it is not able to ensure enough capacity for the data traffic and due to it is not able to ensure data delivering in time. The small communication network capacity can cause data losses as well. The data losses or late data delivery can cause malfunction of the control algorithm and consequently malfunction of the controlled technology.

The algorithm for the design of the electric distribution network was proposed in [9] the algorithms consider not only with installation costs but also operation costs, costs of undelivered electricity and costs that are necessary to pay if the network is not operational (penalty that is paid to the consumers). Moreover, the algorithm considers also a capacity of the grid and necessity of some redundant path in the grid since it is necessary to avoid a disconnection of the grid when some link is disrupted. The algorithm is very contributing for this approach; any previous approach has not considered all of these costs. Therefore, it brings realistic estimation of the costs of the designed network. Unfortunately, the algorithm does not allow setting demanded number of redundant paths for every node pair independently. The redundant paths are created randomly by the genetic operators.

Above described algorithms were chosen as an example of algorithms that are used for the network topology design. Any of the mentioned algorithm is not able to design the network for control engineering. It means network that has different levels of the fault-tolerance in the different parts of the network and thanks to that it can be less expensive than the network with the same level of the fault-tolerance in the whole network. At the same time, the network must allow in time data delivering and prevent to malfunctions of the control systems caused by the late delivering of data or data losses. The algorithm must be able to ensure not only that the average delay does not exceed permitted value but also that no data frame exceeds the maximal permitted delivery delay. Since in control engineering the real hardware is used and often is not possible to expand the number of the communication ports, the algorithm must apply a limitation of the number of the communication ports as well. It is not possible to connect more communication links to the device than there is the number of the communication ports of the device. The algorithm should allow using a priori information of the unsuitable topologies, if it is possible to gain some similar information. The acquisition costs of the network should be as small as possible while the network fulfills mentioned demands.

From time to time, it is necessary to change the network for the control engineering because the controlled technology was changed. Then it is necessary to change the control system very often as well as the network that allows communication of the parts of the control system among each other. The network must also satisfy all demands for the time delay, fault-tolerance and communication ports constrains if the number of nodes must be expanded or reduced. It is necessary to add to the modification costs all costs that are necessary to pay for the network change. It means that the costs for device and cable removal are necessary to consider as well. Moreover, there is a possibility that information about unsuitable topology of the reduced/expanded network is known. Then it is useful to use this kind of information to design the network that satisfies all demands that are laid down on the network. The algorithm that works with all these requirements it not known yet. Therefore, it is necessary to design this kind of the algorithm for an application in the computer systems and especially in the control engineering.

Chapter 4

Network topology design

4.1 Problem formulation

The network is possible to describe as an undirected graph $G = \{V, E\}$, where V is a set of all vertices and E is a set of all edges. The set of all edges in the graph and consequently the set of all communication links should be one of algorithm results. We need to know parameters that are necessary for the network topology design. Since we want to design the network that is as less expensive as possible, we need to know the prices of all possible connections of the nodes in the network. The prices of all possible connections are described in the matrix of acquisition costs C. The matrix C is used during an evaluation of the designed network. The matrix C is symmetrical since the price for connection of two nodes is the same independently on the direction of the communication cable laying (in the real situation this may not be the true, since the cable can be laid in tough environment and the cable laying in one direction can be easier than in another direction). The described cost matrix C is possible to use only if we use one kind of material of communication cables. If we want to use the different materials for the communication cables, one matrix of the acquisition costs will not be able to describe prices of nodes interconnection when different materials are used. Hence, it is necessary to use more matrices of the acquisition costs. Since we expect an application of the metallic and the fibre cables, we must use two different matrices of the acquisition costs. C_F for the fibre cables and C_M for the metallic cables. Every element of the mentioned matrices describes the costs that are necessary to invest for the interconnection of the pair of the nodes. The costs include price of the cable and work that is necessary for the cable laying. Both of them are dependent on the length of the cable, then it is possible to say that the price of the nodes interconnection is a function of the cable length that it is possible to estimate from the position of the nodes in the environment.

Another important parameter for the network physical topology design is the demanded numbers of the independent communication paths between nodes. This attribute is described by the matrix of redundancy M. The matrix M is a symmetrical since the technology, which allows communication in the both direction, is expected to be used. All elements of the matrix M are nonnegative. Element $m_{i,j}$ is 0 if the communication paths between nodes i and j is not needed. It means that the parts of the control algorithm included in nodes i and j do not need to share the data among each other. The bigger element $m_{i,j}$ the more important data are shared between nodes i and j. The actual value of the element $m_{i,j}$ should be chosen very carefully according to the control algorithm and a possibility of the link interruption. From this point of view, the elements should be set as high as possible; from the acquisition costs point of view, the elements should be chosen as small as possible. Hence, it is necessary to balance these points of view regard to natural demands in control engineering: the fault-tolerance is more important than the acquisition costs.

Above mentioned variables are sufficient for the design of the physical topology of the network that is not constrained by the real life request such as the constrained number of communication ports of the nodes or the limitation caused by the environment. Any known algorithm [1], [8], [21], [22], [25], [26], [28], [29], [30] does not consider these limitation and therefore may design the network at which the required number of connections to the node can exceed the number of node communication ports available and consequently makes the network realization impossible (If it is difficult to increase the number of the communication ports). Therefore, it is necessary to have a possibility to inform the algorithm for the network topology design about a number of communication ports at every node. Moreover, if we expect an application of more physical layers than only one, we must store information not only about the number of the communication ports but also about the kind of communication ports of every node. This information is stored in the variable K for every node. This information allows to evaluate every network, which has a bigger number of communication links at some nodes than it is possible, as an inadmissible one since the network is not possible to build because of small number of communication ports at some node. The number and kind of communication ports is very useful to know especially if the network should comprise different kinds of nodes that can have different numbers of communication ports. In this case, it is not possible to use some simple condition that the number of connections at every node must not exceed some certain value which is the same in the whole network. It is possible to use this simple condition if the limitation is set according to the node with the smallest number of the communication ports available. On the other hand, this limitation may prevent a usage of all communication abilities of nodes with bigger number of communication ports than it is this limitation and lead to unnecessary expensive network.

Above mentioned variables allow to design the network physical topology with the different fault-tolerance in the different parts of the network. Moreover, there can be used different physical layers in the whole network and they can be incorporated in the network design without any issue. Nevertheless, the successful design of the inexpensive fault-tolerant physical topology is not sufficient for the design of the network for the control engineering by itself since there is another network attribute important for the network application as well. It is the delay of the data-frame delivering. The inexpensive network is useless if the data are delivered too late and cause malfunction of the control system and consequently malfunction of the controlled technology. The costs connected with the malfunction of the controlled technology can be much bigger than savings gained by the building of inexpensive network that causes this malfunction. Therefore, it is necessary to verify the time of data delivery since it is the only possible variable that can say whether the network has enough performance for the transmission of the expected load.

We must know expected load in the network for a calculation of the data-frame delay. The expected load for every node pairs can be estimated from the knowledge of the behaviour of the control algorithm and communication protocol. We know the amount of data, which are necessary to transmit among nodes, thanks to the knowledge of the control algorithm and its distribution among nodes. The communication protocol determines a communication overhead of every data-frame. These attributes allow estimating overall data amount, which are necessary to transmit among all nodes. The data amounts, which are necessary to transmit among nodes, are stored in the matrix of data-flows F. Every element $f_{i,j}$ of the matrix F corresponds to the data amount that must be transferred between nodes i and j.

There is another important attribute that is necessary for the calculation of the data frame delay. It is a capacity of the communication links in the network. We assume that the communication capacity of all communication links that are made from the same material is the same for those communication links. It means that we have the same number of communication capacities as it is the number of different physical layers.

We can calculate delays of data-frames thanks to the knowledge of the data traffic between nodes but we must to know also the maximal permitted delay of data-frames between nodes.

We can estimate the maximal permitted delay from the attributes of the control system or from the attributes of the control algorithm. A level of sensitivity of the control algorithm to the late data delivery has a crucial importance for the setting of the maximal permitted delay. If the control algorithm is used for a control of a heating system, then the maximal permitted delay can be tens of seconds. However, if the control algorithm is used for the control of the fast servo drives then the maximal permitted delays can be milliseconds and its exceeding can cause serious damage of the controlled technology. Thanks to knowledge of the control algorithm, we can set the maximal permitted delay for every pair of nodes and thanks to that evaluate the delay for every node pair independently on others. This independent evaluation is possible to use during the network topology design, where the number of exceeded permitted delay says how inappropriate the design is. The maximal permitted delays are stored in the matrix of delays D where every elements $d_{i,j}$ corresponds to the maximal permitted delay of data-frames between nodes i and j.

The result of the algorithm exactly describes a placing of the communication links in the network and the acquisition costs of the network as well. The node interconnection is described by the adjacency matrix P where every element $p_{i,j}$ depicts connection of nodes i and j. If $p_{i,j} = 0$ there is no direct connection between nodes i and j. If $p_{i,j} = 1$ there is a direct connection between nodes i and j. The matrix p is symmetrical since we expect that the communication technology allows to communicate in both direction and then if there is a direct connection between nodes i and j there is connection between node j and i as well. The network design gained by our algorithm has the demanded level of the fault-tolerance among sets of nodes and allows to transfer demanded load among nodes while any delay constrains is not exceeded. The network topology has as small acquisition costs as possible at the same time.

The Simple Genetic Algorithm [11] is used for the design of the network physical topology. Genetic operators are described in the following paragraphs.

4.2 Topology with different levels of fault-tolerance

The design of the network physical topology belongs to the NP complete problems [3], [7]. Therefore, it is necessary to use some heuristic method for the solving of the network topology issue [1], [20]. The genetic algorithm was chosen for this purpose. The genetic algorithm cannot ensure a finding of the global optimum but it is able to find a near optimal solution very fast. The big advantage of the genetic algorithm is that its basic version is simple for implementation [12] and the genetic algorithm can very easily implement complex constrains of the solution. The genetic algorithm is very useful at discrete optimization and in condition when optimized function is not continues. Thanks to that, the genetic algorithm is very useful for solution of the network topology design issue. The genetic algorithm brings many issues by its natural attributes and behaviour. The first of them is a representation of the solved problematic. The representation has a huge importance since all genetic operators must work with this representation and they are limited by this representation. Others possible issues are connected to genetic operators and chromosomes evaluation.

4.2.1 Chromosome representation

The chromosome must be able to describe solved problematic and all possibilities that can occur. It is possible to use a description based on the graph theory that works with the adjacency matrix. The adjacency matrix describes node interconnection and thanks to that it exactly says which node is connected to others. We can see in the Fig. 4.1 an example of the network that comprises 5 nodes. The corresponding adjacency matrix is in the Fig. 4.1 as well. We can see that adjacency matrix is symmetrical by its definition [14] (if we expect that the communication in the both direction is possible and the node V_1 is connected with the node V_4 then the node V_4 must be connected with the node V_1). Then we can use this attribute of the adjacency matrix for the network representation very easily because the chromosome can represent only triangular part of the network. If the chromosome represents only a triangular part, the length of the chromosome is reduced as well as the memory amount that is needed for the chromosome storing. The transformation of the adjacency matrix to the chromosome and vice versa is possible to see in the Fig. 4.2.

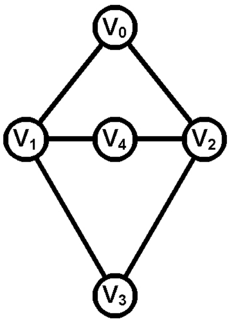

Figure 4.1: Network

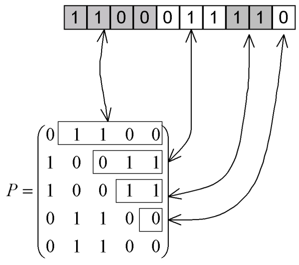

Figure 4.2: Network representation

Then the chromosome is a vector i with l elements:

$$i = \{i_j : i_j \in \{0,1\}\,,, j = 1,...,l\}\,, \tag{4.1}$$

where l is given by:

$$l = \frac{N\,(N-1)}{2}. \tag{4.2}$$

4.2.2 Genetic functions

Genetic functions are the core of the genetic algorithm. They have a direct influence on the speed of a search and results quality. There are two basic genetic operators; both mutation and crossover are based on the natural evolution.

4.2.2.1 Genetic mutation

Genetic mutation operator belongs to the basic genetic operators. The main purpose of the mutation operator is to preserve a diversity of the chromosomes [12]. It is allowed by the small changes that occur at the chromosome population rarely. These "little" changes allow genetic algorithm to search in the whole search space and thanks to that, the genetic algorithm can find a near global optimum solution. Mutation operator prevents stagnation in the local optimum, which is the main disadvantage of the gradient methods. Thanks to the proposed chromosome representation it is possible to use very simple genetic mutation (one bit mutation [12]) which only switch chosen element to another value (if the element is 1, its mutation is 0 and vice versa). The chromosome representation ensures that the mutated chromosome describes only allowable solutions, since the chromosome represents only the triangular part of the adjacency matrix. If the link is removed or added to the network, the link is removed or added not only to the source node but also to destination node (rows of the adjacency matrix represent sources and columns represent destinations). There is an example of the used one bit genetic mutation in the Fig. 4.3.

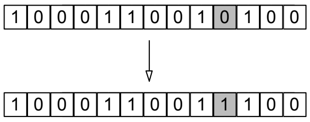

Figure 4.3: Chromosome mutation

4.2.2.2 Crossover operator

A crossover operator tries to improve the genetic population with the help of the previous "good" solutions. The crossover operator combines solutions from the previous iteration to the new solutions in expectation that offspring will be better than parents. The crossover operator is depicted in the Fig. 4.4.

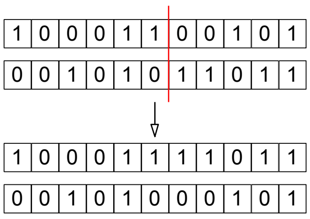

Figure 4.4: Chromosome crossover

It is very easy to imagine that the crossover operator can create impermissible solutions. Fortunately, the chosen chromosome representation always describes permissible solution, since the chromosome describes only the triangular part of the adjacency matrix. Both the mutation and the crossover operator create a new population from the original population and thanks to that, it is possible to discover new solutions of the network topology design issue. The new chromosomes finding would be useless if we were not able to decide which chromosome represents "better" network. Therefore, there must be an evaluative function for every application of the genetic algorithm.

4.2.2.3 Fitness function

The fitness function serves as a chromosome evaluation. There is obvious possibility to use network acquisition costs as a fitness function. Especially, if one of the tasks is a design of as inexpensive network as possible. At this assumption, the acquisition costs of the network can be used as the basic fitness function for the topology evaluation:

$$C_P = C\&P \tag{4.3}$$

The matrix C_P is the cost matrix of the actual network (described by the chromosome that is evaluated); the matrix P is the adjacency matrix of the actual network (described by the chromosome that is evaluated); the operator & is "logical AND" which works as follows $c_{p_{i,j}} = c_{i,j}$ if $p_{i,j} \neq 0$ else $c_{p_{i,j}} = 0$. Since it is much easier to compare one value than the whole matrix, it is possible to count the actual network costs and fitness function according to:

$$c_{ost} = \sum_{i,j=0}^{N} c_{P_{i,j}} \tag{4.4}$$

As one can see in the (4.4), the basic fitness function evaluates only the price of the network and it is not influenced by the fault-tolerance of the network or even by the network interconnections. If we use the basic fitness function and we know that the less expensive network the better network, then the best network according to (4.4) will be the network without any communication links at all and this network definitely does not satisfy demands for the uninterrupted communication. Therefore, it is necessary to use a function that helps to evaluate chromosomes describing network that does not meet the fault-tolerance or other requirements. The function that helps evaluate chromosomes is called the penalty function [11]. The penalty function says that chromosome does not meet our requirements but the chromosome is not automatically excluded from the next generations and may have an influence on them. The chromosome, which does not meet the demands, may be only one mutation or crossover far from the global optimum of the solved task and premature excluding of the chromosome can prevent find the global optimum.

$$c_w = c_{ost} + rPen. \tag{4.5}$$

where

$$Pen = 1 + N^2 c_{\max}. \tag{4.6}$$

Where N is the number of the network nodes; $c_{\max}$ is the maximal element of the cost matrix C and r is the indication that the chromosome should be penalized. If the chromosome does not describe the network which has demanded fault-tolerance in the different parts of the network, then $r = 1$ if the chromosome meets all demands for the different fault-tolerance in the network, else $r = 0$). The equation (4.6) ensures that the chromosome that is penalized has always bigger costs than the chromosome which is not penalized. The equation (4.6) prevents situation when the penalized chromosome has better evaluation than chromosome with many communication links which is not penalized. If the smaller penalty function is used, then the chromosome without any communication links can have better evaluation than the chromosome that meets all demands for the fault-tolerance and connectivity and it must not ever happen.

The equation (4.7) is used as a fast approximate test of the network connectivity and the fault-tolerance.

$$\deg(v) \geq m_{\min} \text{ for } \forall v \in V. \tag{4.7}$$

This equation (4.7) is possible to use only as a fast approximation of the network fault-tolerance and connectivity (see 2 **Reliability**) but it is very fast and can reduce the number of unnecessary chromosome tests. (if there is in the network the node with smaller degree than it is the minimal number of the demanded independent paths, then this network definitely does not meet demands for the fault-tolerance and it is not necessary to use the exact fault-tolerance verification). It is similar to the connectivity verification, if there is a node with $\deg(v) = 0$ then the network has node that is not connected to the rest of the nodes. The equation (4.7) is not able to verify whether the network has the demanded different fault-tolerance in the different parts of the network or if there are disconnected components in the network. Thus, the Ford-Fulkerson [14], [31] algorithm is used for this purpose. The Ford-Fulkerson algorithm allows to verify the number of independent paths for every pair of nodes and the network interconnection is verified during the algorithm initialization. Thanks to that one can be sure that the network which is evaluated as connected with the different demanded fault-tolerance in the different parts of the network really meets requested attributes contrary to the networks gained by algorithms [21], [26], [29], [32]. Unfortunately, the exact verification with the help of the Ford-Fulkerson algorithm needs a lot of time.

Results gained with the help of proposed fitness function can be better than results gained by the fitness function described in [2], [3], [8] since these algorithms do not allow to design the network with as small number of communication links as possible. It can occur if the network can have different levels of the fault-tolerance in the different parts of the network. If the network can have different levels of the fault-tolerance, algorithms that use a node degree as a fault-tolerance measure must design the network that meets demand for the highest node degree in the whole network. It means that these algorithms necessarily design more expensive network than it is possible to design if the demands for the different fault-tolerance are fulfilled exactly. If the network must have a set of nodes which connection is $1 - fault - tolerant$ and some node pairs which interconnection should be $2 - fault - tolerant$, the algorithms which use the node degree as a fault-tolerance measure must design the whole network as $2 - fault - tolerant$. It means, that there must be minimally three independent paths for every node pair contrary to the application of proposed algorithm that is able to design the network containing only necessary links. The proposed algorithm is able to design the network that can meet the demands for the different fault-tolerance exactly thanks to the application of the Ford-Fulkerson algorithm and fitness function (4.7).

4.3 Topology with same level of fault-tolerance

The huge disadvantage of the algorithms [2], [3], [8] is that they are not able to ensure that designed network is really fault-tolerant (see 2 **Reliability**). Therefore, if we really want to design the network with the same level of the fault-tolerance and use a node degree as a fault-tolerance measure we must use a different approach. It is possible to use a degree as a fault-tolerance measure and a fitness function with the penalty for every node that does not have the demanded degree or use a such chromosome representation and genetic operators that allow designing only networks that meet the request for the node degree automatically. Similar technique was developed for the representation of the TSP (Traveling Salesman Problem) [33].

4.3.1 Node degree as fault-tolerance measure

The useful chromosome representation of the TSP issue was developed for the genetic algorithm [33]. The representation use the nature attributes of the TSP: a path passing through the all nodes must create a cycle. Then the issue of the TSP can be easily described as a list of nodes as is depicted in the Fig. 4.5.

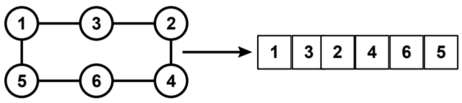

Figure 4.5: TSP chromosome representation

The list of the nodes exactly describes the node interconnection (every node is connected by the communication link directly to its neighbours; the node at the start of the chromosome is connected to the end node and vice versa).

One can see that this representation is simple and very economical at the memory consumption (it needs only 6 memory places instead of 15 places for coding of the adjacency matrix). Moreover, it always ensures that every node has exactly $\deg(v) = 2$ for $\forall v \in V$. Thanks to that, the designed network must be 1-fault-tolerant. This description could be used for the design of the network that should have the same level of the fault-tolerance in the whole network. The genetic operators for solving of the TSP problem are known as well. The application of the described chromosome representation can speed up the design and ensures the 1-fault-tolerance at the same time. Unfortunately, this representation is not possible to use if we need bigger fault-tolerance than one. Therefore, the new solution is proposed and described in the following paragraphs.

The chromosome coding is the same as for the design of the network with different fault-tolerance (Fig. 4.5). It means that the attributes of the adjacency matrix are applied. Unfortunately, there does not exist any attributes that could ensure that the network has demanded fault-tolerance and therefore it is necessary to verify the fault-tolerance exactly or use the node degree (the node degree is possible to use only in the limited cases). It is possible to use the penalty function similar to (4.5). The penalty function would penalize the chromosomes with the different node degree than demanded. On the other hand, another algorithm can create only such solutions that suffice for the demands of the same node degree for all nodes in the network.

The creation of the chromosomes that meet demands for the node degree is not as simple as it seems to be. No exact algorithm that produces this kind of chromosomes is not known. Therefore, a random generation of the initial population is used and after then the repairing function of the initial population is applied. The initial population must contain only chromosomes with the demanded node degree after application of the repairing function.

4.3.2 Algorithm for creation of initial population

Algorithm works with the chromosome but will be described as if it works with the adjacency matrix for illustrative purposes (algorithm works with the chromosome that describes the upper triangular part of the adjacency matrix).

1. Generate chromosome

   Generate genes randomly from the set $\{0, 1\}$. Any node must not have a degree bigger than it is demanded node degree. If it is not possible to set any other element to 1 without exceeding of a constrain to the node degree, verify if the chromosome meets the demands for the node degree at every node. If yes, then satisfying chromosome is generated, continue with the generation of the other new chromosome. If not, continue to step No. 2.

2. Repair chromosome
   (a) Find the first node which does not have the demanded degree (search from the node with the highest index - the lowest row of the adjacency matrix).
   (b) Then continue in searching in the direction to upper rows, find the row, which does not contain 1 in the column corresponding to the node that does not meet demands for the node degree and have 1 at other position.
   (c) Then change elements in the found rows in the following manner: fill in 1 in the column, which corresponds to the row, which does not meet demands for the node degree; if there is 1 at some other column change the first 1 into 0. All operation in this step are done at the row found in the step No.2b. Go to step No.3.
3. Removal of unnecessary links
   (a) If there is some node, which has a bigger node degree than demanded, remove all edges connected to this node.
   (b) If there are steady shifts in the network topology: randomly put 1 at some chromosome element and go to the step No.2. If not, go to the step No.3c
   (c) Go to step 1.

The step No.1 places as many communication links as it is possible without violation of the node degree limitation. If the chromosome does not meet the node degree request, the algorithm repairs the chromosome by the addition and removing of a communication link (step No.2.). The step No.2. does not verify the node degree since the task in the step No.2. was launched because there was not possible to create any other connection in the network without a violation of the node degree constrains. Therefore, it is possible that step No.2. can create a chromosome that exceeds permitted number of the communication links. Thus, algorithm removes all edges connected to the node that has bigger number of the communication links than it is demanded (step No.3.). Then, there are enough possibilities to place other communication links to the network, thanks to the operation at the Step No.2. and 3.

Even, if the communication links are removed and placed at different position, it is possible that there are some kinds of periodic shifts in the topology changes and algorithm does these shifts repeatedly. At that moment, there are two possibilities: discard the whole chromosome and start over with the chromosome design or try to change way of the chromosome reparation. The second possibility is chosen and therefore the reparation procedure goes from the step No.3b. to step No.2. directly in that case. The important question is how to find out that the periodic shifts in the network topology occurred. It is possible to store the last several network changes and compare the new changes with those already done or expect that the chromosome is being changed periodically after certain number of iterations of the repairing algorithm. The number of iterations equal to $100 * N$ was chosen as a condition of periodic shifts (N is the number nodes).

It is possible to see the step No.2 in the Fig. 4.6. The network consists of 6 nodes and every nodes should have degree $\deg(v_i) = 3$ (the network should be 2-fault-tolerant).

$$\left(\begin{array}{cccccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ \boxed{0} & \boxed{0} & \boxed{1} & \boxed{0} & \boxed{0} & \boxed{0} \end{array}\right) \begin{array}{l} \deg(v_0)=3 \\ \deg(v_1)=3 \\ \deg(v_2)=3 \\ \deg(v_3)=3 \\ \deg(v_4)=1 \\ \deg(v_5)=1 \end{array}$$

Figure 4.6: 2FT Repairing operation

In the Fig. 4.6 is an adjacency matrix corresponding to chromosome built in the step No.1. One can see that the node No.4 and No.5 does not meet demands for the node degree ($\deg(v_i) = 3$). Thus, the repairing operations are launched. Step No.2. tries to repair interconnection of the node No.5 to other node (to increase the number of communication links connected to the node No.5). It is possible to see the result of this operation in the Fig. 4.7. The number of communication links at the node 5 is increased but the number of communication link at the node No.3. is smaller than demanded as well as at nodes No.4, 5. The missing communication links (at the node No.3, 4, 5) are added during the next passing through the step No.1 of the repairing algorithm. The elements, which ware changed in the step No.2 of the repairing algorithm, are in the grey boxes.

$$\left(\begin{array}{cccccc} 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & \boxed{0} & 1 & \boxed{1} \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \end{array}\right) \begin{array}{l} \deg(v_0)=3 \\ \deg(v_1)=3 \\ \deg(v_2)=3 \\ \deg(v_3)=2 \\ \deg(v_4)=1 \\ \deg(v_5)=2 \end{array}$$

Figure 4.7: 2FT Repairing operation 2

The way of the chromosome generation and the application of the repairing algorithm ensures that all chromosomes in the initial population have the demanded node degree at the every node (the node degree is the same for all nodes) and therefore they meet the demands for the fault-tolerance. The suitable chromosome representation can help to the genetic algorithm to get better results but it is useless without appropriate genetic operators.

4.3.3 Genetic operators

4.3.3.1 Mutation

The mutation operator for the network with the same demanded node degree for the whole network works similarly as the mutation operator for the TSP. There is only one difference, it is necessary to work with the adjacency matrix instead of elements of the list of the nodes(TSP). In the Fig. 4.8. is the example of the network and corresponding adjacency matrix.

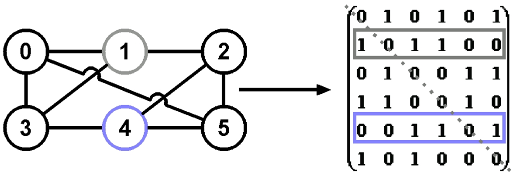

Figure 4.8: 2FT mutation

The network consists of the six nodes and every one has the degree $\deg(v) = 3$. The nodes No. 1 and 4. were randomly chosen to be mutated. The mutation result is depicted in the Fig. 4.9.

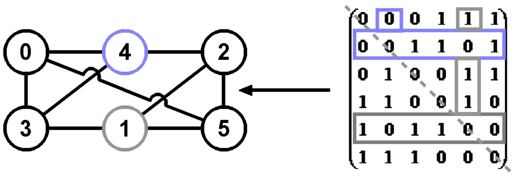

Figure 4.9: 2FT mutation result

Mutation algorithm for n-FT network:

1. Choose the chromosome which is going to be mutated.
2. Randomly choose two nodes to be mutated.
3. Substitute rows, corresponding to the chosen nodes, by each other.
4. Change columns corresponding to the nodes according to the new rows of the adjacency matrix.

5. Verify if all nodes meet the node degree demands; if yes, the chromosome was successfully mutated; if not continue to step No.6.

6. Repair chromosome - Set elements at the coordinates $[Node1, Node2]$ and $[Node2, Node1]$ to 1. Then the chromosome was mutated successfully.

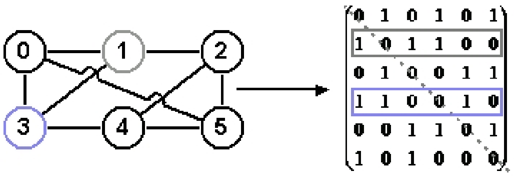

Figure 4.10: 2FT mutation result

In the Fig. 4.10 is a 2-fault-tolerant network containing six nodes. The node No. 1 and 3 were chosen to be mutated. The result of the mutation is possible to see in the Fig. 4.11.

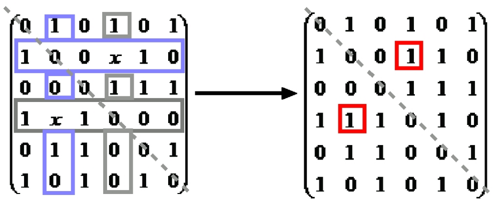

Figure 4.11: 2FT mutation result

It is possible to see at the left part of the Fig. 4.11 that at two positions (elements x) there should be no connection according to the algorithm. If it happens, the rules of the node degree would be violated. Therefore, it is necessary to repair the chromosome; set elements at positions corresponding to the mutated nodes to 1. The reason why it is necessary to repair the chromosome is that at the original gene was 1 at elements that are after mutation exactly at the diagonal of the adjacency matrix and therefore the connection is missing after the mutation. The mutation operator and repairing procedure ensures that all resultant chromosomes meet the demands for the node degree and consequently the network created according to the chromosome has required level of the fault-tolerance.

The mutation operator is not the only one genetic operator, which must be tailored according to the chosen chromosome representation. The crossover operator must be modified as well.

4.3.3.2 Crossover operator

The crossover operator is the important part of the genetic algorithm and allows gaining the new chromosomes from the already existing chromosomes and using part of the information stored in “parents”. The function of the crossover operator is dependent on the chromosome representation. As it was already written, the chromosome representation that describes the triangular part of the adjacency matrix was chosen. This representation and the task of the design of the fault-tolerant network with the same node degree brings by itself an issue of the permissibility of the genetic operator results. Therefore, the chromosomes can be crossed only at places, which are at the borders of the node elements. It allows better manipulation with the data and easier decision if there are some conflicts in the resulting chromosome. The crossover operator crosses the chromosomes at two points and works with two parents.

Crossover algorithm

1. Randomly choose two chromosomes
   (a) Transform the chromosomes to the matrices
   (b) Randomly choose nodes for crossing
2. Crossover nodes
   (a) Swap elements at rows and columns corresponding to the nodes chosen for crossover
3. Verify whether the resulted matrices (chromosomes) meet demands for the demanded node degree
   (a) If both results of the crossing meets demands for the node degree: The results are permissible ones
   (b) If the node degree demand is not fulfilled: Continue with the repairing algorithm.

It is possible to see an example of the crossover operator in the Fig. 4.12.

Two randomly chosen chromosomes should be crossed. The indexes No.3 and No.5 were selected as the borders of the mutation. It means that rows No.3 and No.4 are going to be swapped between the corresponding adjacency matrices. The results of this operation are depicted in the Fig. 4.13. The corresponding columns are exchanged and the rows as well. The consequence of this operations is that the rule of the demanded node degree is violated (some nodes have bigger node degree; other nodes have smaller degree than demanded). Therefore, it is necessary to repair the chromosomes gained by the crossover operator.

In the Fig. 4.13 is possible to see the result of the chromosome reparation. In the red boxes are elements of the matrices that are necessary to repair to fulfill demands

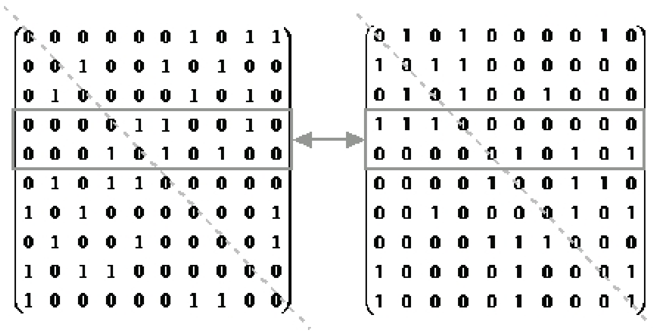

Figure 4.12: n-FT crossover operator

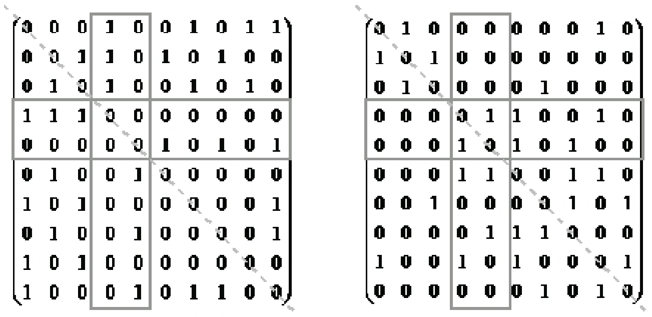

Figure 4.13: n-FT crossover - results

of the node degree. It is necessary to design the new part of the network topology in these boxes and elements outside of these boxes give us the constraints for this repair.

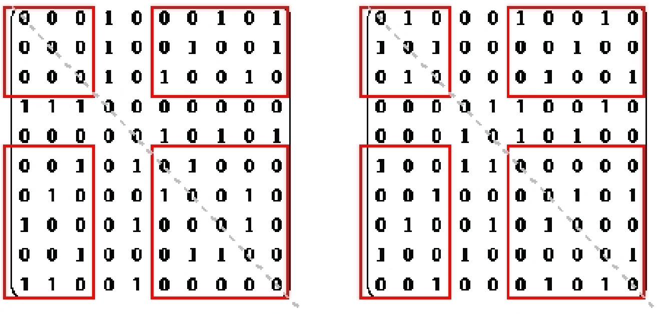

Figure 4.14: n-FT crossover - after reparation

Reparation algorithm after the crossover operation:

1. Remove elements at rows which violate the rules of the demanded node degree (remove all elements excluding those which correspond to the rows that have been crossed).

2. Find the row with the smallest number of non-zero elements. (Find the nodes with the smallest node degree). Continue to the step No.3.

3. Try to put 1 at the row, which have been found at the step No.2.

   (a) Find the column, which corresponds to the node that has a smaller degree than demanded.

   (b) Put 1 if in the found column was 0 in the matrix from which have been taken the crossing element. Continue to the step No. 3c.

   (c) If you have found the column but there have been 1 at the corresponding element of the matrix from which have been taken the crossover part, then continue to the step No.3d.

   (d) If the row found at the step No.2. have the demanded degree or all possibilities for placing of 1 was already tried unsuccessfully, continue to the step No.4. otherwise go to the step No.3a.

4. If there is some row, which does not meet demands for the node degree and it was not already tried to be repaired, go to the step No.2. If all possible rows were already tried to repair go to the step No.5.

5. If the matrix meets demands for the node degree, then the chromosome was repaired successfully and it is possible to continue in the same way with the other results of crossover operation. If not, continue to the step No.6.

6. Do the same as in the step No.2., go to the step No. 7.

7. Try to put 1 at the row, which was found at the step No.6.

   (a) Find the column that corresponds to the node that has a smaller degree than demanded and put 1 there.

   (b) If the node found in the step No.6. does not meet demand for the node degree and not all possible places of 1 have been tested yet, go to the step No.7a. Otherwise, continue to step No.8.

8. If not all rows were repaired and all rows were not already tried to be repaired, go to the step No.6. Otherwise, go to the step No.9.

9. If the all nodes meet demand for the node degree, then the reparation was successful and one can continue in reparation of other results of the crossover operation. Otherwise, the reparation was not successful, return the original matrix and then continue with the reparation of other results of the crossover operation.

The reparation algorithm works as it is described above. It is not very likely that the resultant matrices meet demands for the node degree after the application of the crossover operator. Therefore, it is necessary to launch the reparation operation. The Step No.3. allows finding different topology than is described by only one of the parents (see the step No.3c). This allows preserving part of the information of both parents and doing only necessary repairs in order to meet the node degree demand. If it is not possible to repair it in that way, another procedure tries to repair the chromosome in way that the network topology can be almost the same as one of the parents (see the step No.6.).

It is possible to see see an example of the networks and crossover operators in the Fig. 4.15, 4.16. The networks are the same as there are described by the example of

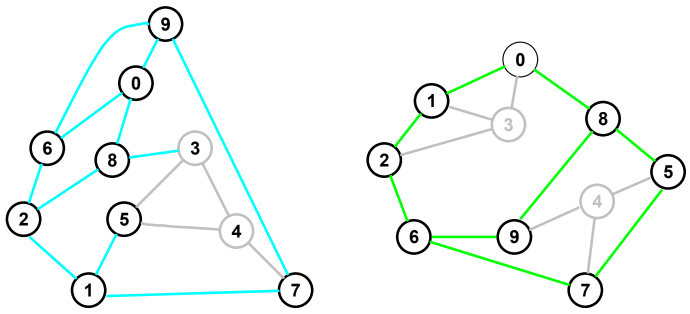

Figure 4.15: n-FT crossover - network

the reparation algorithm (see the adjacency matrices). The nodes No.3. and 4. were chosen to be mutated (grey colour). It means that also the connections to other nodes from the chosen nodes should be preserved. One can see the results of the crossover and repair operation in the Fig. 4.16.

The communication links, which were added by the reparation algorithm, are in the black colour. Link and nodes, which were mutated, are in the grey and link which are the same as some links of the original networks (before crossover) have the same colour as was their colour in the original network. It is possible to see that the network on the left side of the Fig. 4.16 has only a few communication links that are the same as those in the original networks (the crossover part was remained and only two communication links from the parents as well). Nevertheless, the parts of the topology from the original networks were preserved and new parts were created only in order to meet demands for the degree of the nodes. The resultant network on the right side of the Fig. 4.16 shows that the crossover operator is able to preserve most of

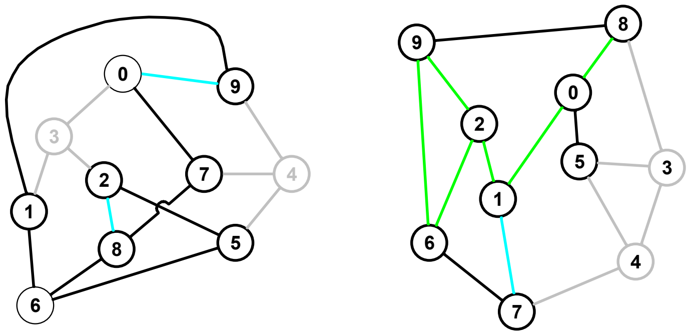

Figure 4.16: n-FT crossover - network - after reparation

the structure of the original networks. The resultant network is mostly a combination of the original networks.

Generally, the crossover operator should combine information from the successful chromosomes of the previous generation in order to improve the following generation. This natural attribute is fulfilled in this case thanks to the proposed reparation algorithm that it is necessary to use since the crossover operator by itself often generates the chromosomes that do not meet demands for the size of the node degree. The reparation algorithm allows gaining the chromosomes that combine the original topologies of the parents and meet demands for the node degree at the same time.

4.3.3.3 Fitness function

The important part of the genetic algorithm is a fitness function since it allows to evaluate a quality of the chromosome and consequently a quality of the topology as well. The acquisition costs can be very useful evaluative measure for the design of the fault-tolerant network that should have the same level of the fault-tolerance in the whole network.

It is possible to assume that the acquisition costs include all costs that are necessary for the creation of the actual network. It means that the acquisitions costs should include not only the cost of communication cables but also the cost of work which is necessary to do as well as the costs of the additional communication interfaces (if they are needed) and so on. Then the evaluative function and the acquisition costs are the same:

$$c_{ost} = \sum_{i,j=0}^{N} c_{P i,j}. \tag{4.8}$$

Where $c_{P i,j}$ are elements of the matrix C_P, which is the matrix of the acquisition costs of the designed network and exactly describes the price of every necessary communication link and it is possible to gain this matrix according to following equation:

$$C_P = C \& P. \tag{4.9}$$

The matrix C is the cost matrix for the actual network (network described by the evaluated chromosome). And Operator & is “logical AND” which works as follows $c_{p_{i,j}} = c_{i,j}$ if $p_{i,j} \neq 0$ else $c_{p_{i,j}} = 0$.

The best network topology is not with the highest evaluation but the one with the smallest since the acquisition costs are used as a fitness function.

It is not necessary to employ any penalty function in the fitness function since all chromosomes are permissible and it is possible to create the real network according to their structure. The repairing functions at the crossover and mutation operator are able to ensure that chromosomes are permissible as well as chromosomes created for the initial population. Thanks to that it is not necessary to consider any penalty function and use only the fitness function described by the equation (4.8).

4.4 Logical topology design

The communication network for control engineering has specific demands for the delay of the data delivering since the control algorithm can be very sensitive for the late data delivery. If the control algorithm receives data too late, then the control algorithm can decide wrongly and consequently cause a malfunction of the controlled system. Thus, in time data delivery is a crucial for the correct function of the control system and the network must allow in time data delivering. There are two possibilities how to find whether the network allows in time data delivering: it is possible to build the network and after then hope that the network has enough capacity to handle the network load or verify the network behaviour before the actual build of the network. The first choice is not an option since it could be very expensive to build the network and after then verify whether the network allows to transmit demanded network load and repair it if the network would not be able to transport the demanded load. Therefore, it is necessary to use the simulation of the network behaviour before the actual build of the network.

We need to know not only the network physical topology but also the logical topology of the network for the verification of the time of the data delivery. It means that one must design also the logical topology for the verification purposes. **The logical topology says which communication path is used for the data sending. It does not say anything about a schedule according which data are sent or behaviour of the communication network if the data are not sent or collide with other data.** These issues are solved by the communication protocol or by the scheduling algorithm [34]. The genetic algorithm is used for the design of the logical topology.

It is crucial to tailor the genetic algorithm exactly to the task, which the algorithm should solve. Therefore, it is necessary to design the algorithm very carefully according to the task that should be solved and try to simplify the task as much as it is possible. The simplification is important since the algorithm works with a huge search space and it needs a lot of time for the solution finding. Therefore, the simplification can be very useful.

The algorithm for the network topology design is forced to find the topology, which has several independent communication paths, by the demands for the network fault-tolerance. If the algorithm that is able to design only the network with the same level of the fault-tolerance in the network is used, there must be minimally one redundant path for every node pairs. Therefore, the topology of the network can be quite complex and the search space can be vast. Unfortunately, if there is a redundant communication path it is hardly possible to do some easy simplification of the network contrary to the network where the network part with the bus topology exists. However, this kind of the network can never be designed by the algorithm, which designs the network with the same level of the fault-tolerance in the whole network. Nevertheless, the proposed algorithm for the design of the physical topology with the different levels of the fault-tolerance can design the network topology, which includes the bus topology, if there are not demands for the fault-tolerance at some nodes.

It is possible to see an example of the network containing the bus topology in the Fig 4.17. It is easy to imagine that the logical topology is exactly determined by the physical topology in the part with the bus physical topology.

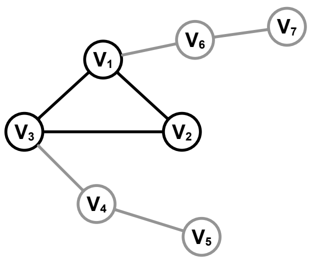

Figure 4.17: The network

There is an example of the network with the physical topology containing the bus physical topology in the Fig 4.17. The network is possible to divide into two parts, the “core” of the network and the “branches” of the network. The core of the network is a part in which is possible to change the logical topology without changes of the physical topology. It means that there must be redundant communication paths in the “core” of the network, since it is not possible to change the logical topology without any redundant communication path. The “core” of the network depicted in the Fig 4.17 contains the nodes V_1, V_2 and V_3.

On the other hand, the branches contain only one communication path and therefore the logical topology is strictly described by the physical topology and can not be changed without changes of the physical topology (it is possible to see in the Fig 4.17; the “branches” of the network contain nodes V_4, V_5 and V_6, V_7). Therefore, the logical topology in the branches cannot be designed in other way than the physical topology allows and there is not any possibility to change the way how the data are exchanged.

Thus, it is not possible to design the logical topology in the branches differently and this attribute can be used for reduction of the search space of the task of the logical topology design.

It is possible to assume that the network is connected and there is any unconnected node during the design of the logical topology. If this assumption is not fulfilled, the design of the logical topology is useless, since we expect that it is possible to design the logical topology and the nodes have uninterrupted communication paths among them.

The algorithm of the branch finding:

1. Find the node with the degree $\deg(v_i) = 1$
2. Find the node connected to the node found in the previous step.
3. If the node has $\deg(v_i) \geq 3$, it is the "end" node of the branch, go to the step No.4. If the node has $\deg(v_i) = 2$, go to the step No.2. If the node has $\deg(v_i) = 1$, then the whole physical topology is connected like a bus and therefore it cannot be optimized anymore.
4. Save all nodes of the branch and continue to the step No.1, if all nodes with the $\deg(v_i) = 1$ have not been found as a part of some branch.

The network simplification provided according to the algorithm described above decreases the size of the search space of the logical topology design and thanks to that, it speeds up the design of the network topology. The important question is whether the part of the network topology, which has been excluded from the logical topology design, has some influence on the part in which it is possible to optimize the logical topology. It is possible, there are no data-flows from the branches to the core of the network and vice versa, but generally this case is very rarely and therefore there must be a way in the logical topology design how the data-flows in the branches can influence data-flows in the network core. If one omits this influence, which branches definitely have on the network core, the logical topology in the network core would be designed wrongly and the network simplification would do the design of the network topology useless, since the verification of data-flow delays would be wrong and gives us completely wrong description of the network behaviour under the demanded load. Therefore, it is necessary to know how the data-flows in the branches influence the data-flows in the network core. The relation between the communication link capacity, network load and the delay of the data-flow must be known for this purpose.

4.4.1 Data-flow delay calculation

The data-flow delay calculation allows to find the delay of the data-flow if the capacity of the communication link and its load is known. We expect the application of the network in control engineering where the amount of the communication technology based on the packet-switching network still grows. The packet-switching technology is used at the Ethernet technology. The Ethernet technology is used at industrial communication protocols as *Profinet®, Powerlink®, EtherCat®*, etc. The Ethernet technology allows connection of the common network traffic with the traffic used for the control technology purposes. The basic model for the packet-switching network was developed for the ARPANET network. The average delay in the network is possible to calculate by the Kleinrock's interpolation [19].

$$T = \frac{1}{\gamma} \sum_{i=1}^{k} \frac{f_i}{c_i - f_i} \tag{4.10}$$

Where f_i is load of the communication link; c_i is the capacity of the communication link; γ is total arrival rate into the network and k is the number of the communication link. f_i, c_i and γ are in bits per second. This interpolation is possible to use also for the calculation of the delay at every link not only for the calculation of the delay of the network. The (4.10) is possible to use only if the condition $f_i \leq c_i$ is valid. The Kleinrock's interpolation is widely used for an estimation of the network behaviour. The interpolation is accurate enough if certain assumptions are fulfilled. The assumptions are as follows: The Poisson's distribution of the arrival time, exponential packet length distribution, infinite nodal storage front, fixed routing, error free communication channels, no nodal delay and independence between transmission time and interarrivel time [19].

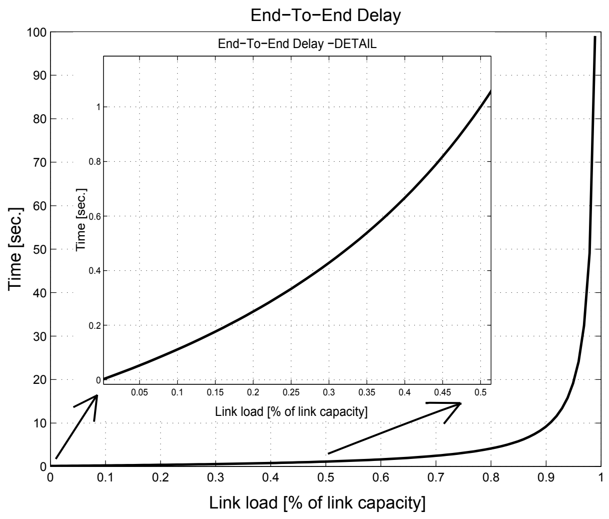

Figure 4.18: End-to-end delay according to link load

The dependence of the end-to-end delay and the load of the communication link is depicted in the Fig. 4.18. It is possible to see that if the network load approaches the link capacity, the delay grows to infinity. The network load in control engineering uses small part of the network capacity mostly. Therefore, it is possible to reach small end-to-end delays that are almost linearly dependent on the communication load.

Thanks to knowledge of the network behaviour, it is possible to simplify the network since is possible to find out the delays in the branches caused by the network load. This knowledge allows to transform the matrices of delays and flows of the network core and thanks to that to decrease the size of the search space.

It is possible to see it at the following example. The same network as is depicted in the Fig. 4.17 is expected. The network consists of the core containing the nodes V_1, V_2 and V_3 and the branches containing nodes V_4, V_5 and V_6, V_7. The matrix of the expected flows looks like:

$$F = \begin{pmatrix} 0 & f_{1,2} & f_{1,3} & f_{1,4} & 0 & 0 & f_{1,7} \\ f_{2,1} & 0 & f_{2,3} & 0 & f_{2,5} & f_{2,6} & 0 \\ f_{3,1} & f_{3,2} & 0 & f_{3,4} & 0 & 0 & 0 \\ f_{4,1} & 0 & 0 & 0 & f_{4,5} & 0 & f_{4,7} \\ 0 & f_{5,2} & 0 & f_{5,4} & 0 & f_{5,6} & 0 \\ 0 & 0 & f_{6,3} & 0 & f_{6,5} & 0 & f_{6,7} \\ f_{7,1} & 0 & 0 & f_{7,4} & 0 & f_{7,6} & 0 \end{pmatrix}. \tag{4.11}$$

The matrix of the maximal permitted delay is:

$$D = \begin{pmatrix} 0 & d_{1,2} & d_{1,3} & d_{1,4} & 0 & 0 & d_{1,7} \\ d_{2,1} & 0 & d_{2,3} & 0 & d_{2,5} & d_{2,6} & 0 \\ d_{3,1} & d_{3,2} & 0 & d_{3,4} & 0 & 0 & 0 \\ d_{4,1} & 0 & 0 & 0 & d_{4,5} & 0 & d_{4,7} \\ 0 & d_{5,2} & 0 & d_{5,4} & 0 & d_{5,6} & 0 \\ 0 & 0 & d_{6,3} & 0 & d_{6,5} & 0 & d_{6,7} \\ d_{7,1} & 0 & 0 & d_{7,4} & 0 & d_{7,6} & 0 \end{pmatrix}. \tag{4.12}$$

The network is changed only into the core after the application of the branch removal. The nodes in the core are going to be transmitters and receivers not only for data-flows generated in the network “core” but also for the data-flows generated and received in the branches. The matrix of flows and matrix of the maximal permitted delay is changed due to the branch removal algorithm as well. The new matrices of flow looks like:

$$F_n = \begin{pmatrix} 0 & f_{n1,2} & f_{n1,3} \\ f_{2,1} & 0 & f_{2,3} \\ f_{n3,1} & f_{n3,2} & 0 \end{pmatrix}. \tag{4.13}$$

Where the new matrix of flows does not only contain the original flows generated in the core but also the flows going from the branches to the inside of the “core” or only passing through the “core” to another branch. The nodes in the “core” are the new inputs or outputs node for the data-flows going from or to the branches. It means that flows in the core looks like:

$$\begin{array}{l} f_{n1,2} = \{f_{1,2}\} \\ f_{n1,3} = \{f_{1,3}, f_{6,3}, f_{6,5}, f_{7,4}\} \\ f_{n3,1} = \{f_{3,1}, f_{4,1}, f_{4,7}, f_{5,6}\} \\ f_{n3,2} = \{f_{3,2}, f_{5,2}\} \end{array}. \tag{4.14}$$

We can see that the network simplification reduces the number of flows for which it is necessary to design the logical topology from 23 to 13. Nevertheless, before the start of the logical topology design it is necessary to apply these changes of the flows into the matrix of the maximal permitted delay, which looks like this:

$$D_n = \begin{pmatrix} 0 & d_{1,2} & d_{1,3} & d_{n1,4} & 0 & 0 & d_{1,7} \\ d_{2,1} & 0 & d_{2,3} & 0 & d_{n2,5} & d_{n2,6} & 0 \\ d_{3,1} & d_{3,2} & 0 & d_{n3,4} & 0 & 0 & 0 \\ d_{n4,1} & 0 & 0 & 0 & d_{4,5} & 0 & d_{n4,7} \\ 0 & d_{n5,2} & 0 & d_{5,4} & 0 & d_{n5,6} & 0 \\ 0 & 0 & d_{n6,3} & 0 & d_{n6,5} & 0 & d_{6,7} \\ d_{n7,1} & 0 & 0 & d_{n7,4} & 0 & d_{7,6} & 0 \end{pmatrix}. \tag{4.15}$$

The matrix has still the same structure as the original one since it is necessary to verify the delay of all data-flows but the elements are changed. The elements corresponding to the delays of the data-flows created and received in the “core” of the network are without any change but the elements corresponding to the data-flows created or received in the branches and going through the network “core” are changed. These elements are smaller because they are reduced by the delay caused in the branches. It means that it is necessary to count the delays for the data-flows in the branches before the matrix of the maximal permitted delay (4.12) is changed into the matrix of the maximal permitted delay for the reduced network (4.15). The delay in the branches is calculated according to (4.10). Then, the new maximal permitted delay looks like:

$$d_{ni,j} = d_{i,j} - d_{Bi,j}. \tag{4.16}$$

Where $d_{Bi,j}$ is a sum of all delay in the branches for the data-flow $f_{i,j}$.

If there are the branches in the network, the knowledge of the delay in the branches is important from other reason as well. The knowledge of the delay in the branches can say whether is possible to design the logical topology of the network.

$$d_{ni,j} \leq 0 \tag{4.17}$$

If the equation (4.17) is valid, it is impossible to design the logical topology without a violation of the physical law (the data-flow transport needs some time and therefore it can not be zero or even negative). If this happen, it is not possible to design the logical topology without the changes of the physical topology and it means that it is necessary to design the physical topology of the network again from the very beginning.

4.4.2 Worst-case scenario

The behaviour of the network has a stochastic character since the data-flow transfer is a stochastic process (provided that there is no network traffic schedule, according to the traffic would be sent through the network in the precisely given time. The data-flow schedule ensures that any data-flow does not have to wait in some queue of the node receiver till the node can accept the data-flow and the data-flow can go directly to the node without any addition delay caused by the waiting in the receiver node).

It is difficult to estimate the network behaviour if there is not the network schedule, therefore it is necessary to define the boundaries of the network behaviour. It is possible to expect that there are not constraints for the early data delivering but there are the constraints for the late data delivery. Therefore, it is important to find the maximal possible delay of data delivery - the worst-case delay.

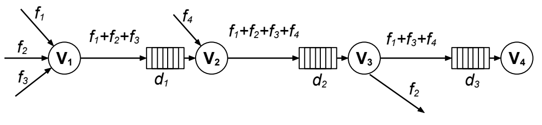

Figure 4.19: Worst-case scenario

In the Fig. 4.19 is an example of data-traffic between nodes V_1 and V_4. It is possible to see the worst-case scenario for the data-flow f_1. The worst situation (excluding the link interruption) occurs if all other data-flows are sent before the data-flow f_1. Especially, if the all data-flows are sent as one burst in which the data-flow f_1 is at the end of the burst. It means that between nodes V_1 and V_2 the data-flows f_1, f_2 and f_3 are sent as one burst of the data and the data-flow f_1 is at the tail of the data-burst. The data-flow f_4 comes to the node V_2 and it is added to the incoming data-burst from the node V_1. The data-flow f_1 is still at the tail of the data-burst. Then, the data-flow f_2 leaves the node V_3 via a different communication link and only data-flows f_1, f_3 and f_4 continue to the node V_4 (the data-flow is still at the tail of the data-burst).

Then the load for the communication link between nodes V_1 and V_2 is $f_1+f_2+f_3$, the load for the communication link between nodes V_2 and V_3 is $f_1+f_2+f_3+f_4$ and the load for the communication link between nodes V_3 and V_4 is $f_1+f_3+f_4$. These loads must pass through the queue at every node and this pass causes the transport delays d_1, d_2 and d_3 that can be calculate from the (4.10).

Then the delay for the worst-case scenario for the data-flow f_1 is the sum of the all delays on the communication path from the transmitter to the receiver node. The worst-case delay for every data-flow is possible to calculate in the similar manner. The data-flow for which the delay is calculated is at the tail of the data-burst that transmits all data-flows between two nodes. The result of this calculation is the maximal possible delay for the data-flow and only a few data-flows can have this delay at the same time in the whole network.

It is **not possible** that the real delay for the data-flows f_1 **and** f_3 would **correspond to the worst-case scenario delays at the same time** since it is not possible to have both data-flows f_1 and f_3 at the tail of the data-burst. Thus, the worst-case delay calculation is highly pessimistic and does not describe the network behaviour realistically but gives us the upper bound of the data-flows delay. There is no way, how could be this upper bound of the network delay violated (excluding the link interruption). Therefore, if the worst-case delay is smaller than maximal permitted delay one can be sure that the network allows transmitting of the demanded traffic load without the violation the maximal permitted delay in any possible situation. Thus, the worst-case delay is useful for the network behaviour description.

4.4.3 Logical topology representation

The important part of the genetic algorithm is a representation of the solved task. The appropriate representation can save a lot of trouble during the logical topology design. Therefore, it is necessary to tailor the representation of the task, which is solved by this algorithm, to the nature attributes of the solved task. It is necessary to describe the logical topology of the network in this case. The logical topology says which communication links are used for the transmission of every data-flow. It means that one must be able to describe every possible communication path for every data-flow of the network. In the Fig. 4.20 is the "core" of the network depicted in the Fig. 4.17.

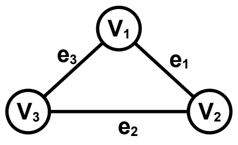

Figure 4.20: The Core

The natural and the easiest description of the logical topology is to use the coding of the edges of the network for this purpose. If it is possible to expect that the physical topology of the network is already given and it is not changing in time, then the number of the communication links is stable and it is not changing in time as well. Therefore, it is possible to describe a path that must be used by the data-flow from the transmitter to the receiver node. It is possible to see this at the following example.

There are two data-flows $f_{1,2}$ and $f_{2,3}$ for which the logical topology should be found. The flow $f_{1,2}$ can use a communication link e_1 or communication links e_2 and e_3, the flow $f_{2,3}$ can use a communication link e_2 or communication links e_1 and e_3. It is possible to see that the chromosome must be able to describe all communications links in the network core as a part of the communication path for every communication link. It means that for the description of the communication path of one data-flow it is necessary to use the same number of elements as is the number of the communication links in the network core. Then for the description of the whole logical topology for all data-flows is necessary to have l elements.

$$l = jm \tag{4.18}$$

Where l is the length of the chromosome for the logical topology description, m is the number of the communication links in the core and j is the number of the data-flows in the network core.

4.4.4 Generation of initial population for logical topology design

The important part of the genetic algorithm is a generation of the initial population. The species in the population should be enough diverse. Therefore, the chromosomes in the population should be as random as possible. The generation randomness brings an issue of the chromosome permissibility. It is easy to imagine that the random generation of the chromosome can generate such species, which does not describe a permissible logical topology at all. The randomly generated gene can describe the communication path that is several times disconnected or even does not start or end at the nodes that should be the transmitter or the receiver node for that data-flow. Therefore, it is necessary to find a different kind of the chromosome generation than pure random generation.

If we look in the Fig. 4.20 we can see that we are able to find all possible communication paths for every data-flow. If we use the Ford-Fulkerson algorithm [14] for the all node-pairs we gain the set of all possible communication paths for data-flows. This set of all possible paths allows to generate the chromosome randomly.

Algorithm

1. Create set of all possible communication paths for every data-flow.

2. For every data-flow randomly choose one possible communication path and place it in the chromosome.

It is possible to show the algorithm at the following example. There is the same network as it is depicted in the Fig. 4.20 and algorithm should generate the chromosome which describes the communication path for data-flow $f_{1,2}$ and $f_{2,3}$. The MSB (Most Significant Bit) corresponds to the edge e_3 and LSB (Least Significant Bit) corresponds to the edge e_1. Then the set of all possible paths for the flow $f_{1,2}$ contains elements $\{001, 110\}$ the set of all possible communication paths for data-flow $f_{2,3}$ contains elements $\{010, 101\}$. The chromosome describing the logical topology for these data-flows is randomly chosen from these two sets of all possible communication paths. The chromosome could be e.g.

It is possible to create the whole initial population in the similar manner.

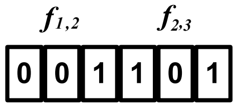

Figure 4.21: The chromosome - logical topology

4.4.5 Genetic operators for the logical topology design

Genetic operators must correspond to the chromosome representation of the solved task. Therefore, it is necessary to modify the genetic operators to the chromosome representation. The network depicted in the Fig. 4.22 is used at the following paragraphs for easier understanding.

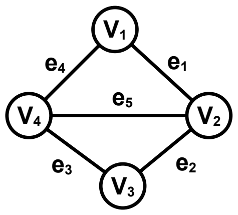

Figure 4.22: The network - logical topology

The chromosome will describe the logical path only for two data-flows $f_{1,3}$ and $f_{2,4}$. Then the set of all possible communication paths for the data-flow $f_{1,3}$ is $\{00011, 01100\}$. The set of all communication paths for data-flow $f_{2,4}$ is $\{10000, 01001, 00110\}$. The communication link e_5 corresponds to the MSB and e_1 to LSB.

4.4.5.1 Mutation genetic operator

The “one bit” mutation operator is used for this purpose. The operator must be modified since the real one-bit mutation [12] can cause that the result of the mutation is inadmissible because the communication path described by the chromosome is wrong. Therefore, the mutation operator can mutate only complete logical path for chosen data-flow. It is necessary to know the set of all possible logical paths for every data-flow for this purpose.

The common one-bit mutation [12] randomly chooses the bit for the mutation and change it over. Similar technique is possible to use if the physical topology has the ring structure (one value - the flow goes in one direction, another value - the flow goes in another direction; mutation is possible only between these two values). If the physical topology has different structure, the technique is not such straightforward, since there are more possible values in which the mutated element can be changed over.

The application of the mutation operator is explained at the following example for the chromosome describing the logical path for the data-flow $f_{1,3}$ and $f_{2,4}$ of the network depicted in the 4.23.

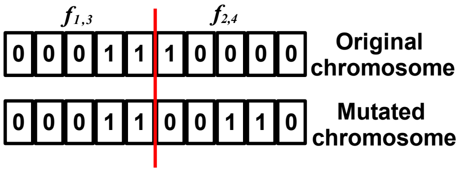

Figure 4.23: Logical topology - mutation

The original chromosome describes the logical topology as $\{00011, 10000\}$ the "bit" chosen for the mutation is the "bit" describing the logical path for the data-flow $f_{2,4}$. The mutation must change the whole "bit" describing the logical path not only a part of the logical path; if so, the gene would describe path that is not possible to use because of its inadmissibility. Therefore, knowledge of the set of all possible paths for every data-flow is used and the result of mutation must correspond to one of the elements from the set for the chosen data-flow. The element, which substitutes the original one is chosen from the rest of the set of all possible paths for data-flow $f_{2,4}$ $\{01001, 00110\}$. In this case, the element $\{00110\}$ was chosen and the original chromosome was mutated into $\{00011, 00110\}$. The procedure of the randomly chosen element from the set of all possible paths is possible to use for every data-flow and the number of the possible communication path does not limit it.

4.4.5.2 Crossover operator

The one point crossover operator is used for the logical topology design purposes. The crossover operator works so that it randomly chooses the place of the crossing and chromosomes are crossed at that place. It is important to choose the appropriate place of the crossing. The place is chosen randomly but its position must be between two elements describing the logical path of the data-flow. If the position of the crossing is placed somewhere in the middle of the element describing the logical path, the result of the crossover operation will be impermissible.

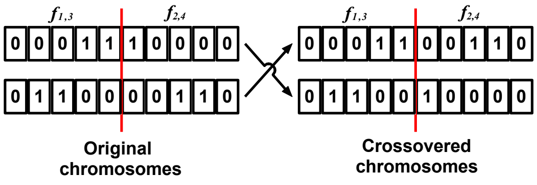

Figure 4.24: Logical topology - crossover

If the position of the crossover is placed between two elements describing the logical path the results of the crossover operations are always permissible, since all chromosomes in the population are permissible (see generation of the initial population, mutation operator). Therefore, if the crossover operation does not create the impermissible chromosomes, there is any other way how the impermissible chromosomes could occur in the chromosome population.

4.4.5.3 Fitness function

The fitness function allows evaluation of the quality of the chromosome and chromosomes population as well. It is important to design the fitness function according to solved task. In this case the task is to design the logical topology with the smallest delay of data-frames as possible. The logical topology must allows in time delivery of all data-frames since every data-frame has its own maximal permitted delay of the delivering and this boundary must not be exceeded. The delay of the data-flows in the core is easy to calculate since the delay in the network branches is already known. The delays in the branches were calculated according to the algorithm (see 4.4.1). If the constrain (4.17) is not valid and therefore it is possible to design the logical topology in the network core. It is not necessary to consider the delay in the branches because the delay in the branches is the same for every chromosome (the delay in the branches have already influenced the maximal permitted delay in the matrix (4.15) that describes the maximal delays after branches removal). Therefore, it does not have an influence on the quality of the chromosome (the influence is the same for all chromosomes).

The delay of the data-frames in the network is possible to calculate according to (4.10). The sum or the average value of the delay can describe the quality of the chromosome. The delay is smaller, the chromosome is better. The delay is bigger, the chromosome is worse.

$$Fit_l = \sum_{i=1}^{j} d_{Fi} + Pen_L \tag{4.19}$$

d_{Fi} is the worst-case delay of the data-frame i, Pen_L is a chromosome penalization during the logical topology design.

The delay is able to describe the chromosome quality good enough but it is not able to describe whether some constrain of the maximal permitted delay is violated or not. Therefore, it is necessary to add a penalty function to the fitness function, The penalty function must be able to ensure that every chromosome which fulfills demands for the maximal permitted delay at every data-flow has better evaluation than the chromosome which violates some of these demands. Moreover, the probability that the designed logical topology does not allow to satisfy the demands at more than one data-flow is quite high at the start of the evolution. Therefore, it is useful to recognize the number of violated constrains and to have different penalty for the different number of the violated demands. Thus, the number of violation has direct influence on the size of the penalty:

$$Pen_L = k_D \max(D) N^2. \tag{4.20}$$

Where $\max(D)$ is the maximal element of the matrix of the maximal permitted delays, N is the number of nodes in the network and k_D is the number of violation of the maximal permitted delay constraints.

The application of the different penalty for the different levels of the fulfilled demands for the maximal permitted delays allows to not lose the information from the chromosome which does not meet demands for the maximal permitted delay. Thanks to that, information included in these chromosomes can have an influence on the chromosomes in the next generation even if their quality is not very good. The information in these chromosomes can improve next generations of the chromosome thanks to the genetic operators.

4.4.5.4 Selection and end condition

The tournament selection is used for a selection of the chromosomes for the next generation. Moreover, the elitism mechanism is applied. Thanks to that, the best chromosome in the population is always preserved to the next generation and can take a part in the population evolution. The chromosome set, at which the selection mechanism is applied, contains not only chromosomes of the actual generation but also the results from the genetic operations. The tournament selection randomly chooses chromosomes form this set to the next generation. It means that the offspring can replace its parent even if the offspring quality is worse (provided that the parent is compared with the better chromosome and the offspring is compared with the worse chromosome).

The number of iteration is applied as the end condition for the logical topology design. The design is stopped after a hundred iterations and the best chromosome is understood as the result of the logical topology design. The resultant chromosome describes the logical topology of the network and its fitness says the delay of the all data-flows in the network in the worst-case scenario.

4.5 Algorithm for network design

The design of the network topology is a complex task and it is necessary to consider many possibilities of the design. The network must meet demands for the different fault-tolerances and allows transferring of the demanded data-load in time. Moreover, the network should be as less expensive as possible.

It is important to preserve the design of the logical topology split from the design of the network physical topology since this division allows to use algorithm modularity. The algorithm modularity is useful since there is not any preferred communication technology in control engineering and therefore the algorithm should allow easy modification for the different communication technologies. The design of the network physical topology is by itself useless since it does not say anything about the network ability to transfer demanded load and the logical topology design needs the network physical topology as one of the inputs. Therefore, it is necessary to bide this parts of the network design together.

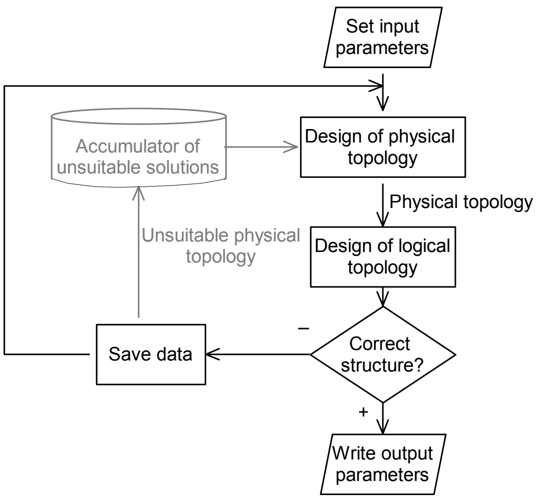
Set input parameters
Accumulator of unsuitable solutions
Design of physical topology
Physical topology
Unsuitable physical topology
Design of logical topology
–
Save data
Correct structure?
+
Write output parameters

Figure 4.25: Algorithm - network topology design

The whole algorithm is depicted in the Fig. 4.25. It is possible to see a binding of the design of the physical and the logical topology. Both, the logical topology design as well as the physical topology design are important for the design of the network for control engineering and therefore they must share the results. The whole algorithm works as follows:

1. In the first step, it is necessary to *Set* all *input parameters*. It means that all data are set; the matrix of the maximal permitted delay, the matrix of the data-flows, the matrix of fault-tolerance, the matrix of the acquisitions costs. The accumulator of the inappropriate solutions is cleared.

2. In the step: *Design of physical topology*, the physical topology is created. The algorithm is simple genetic algorithm [12] that uses a chromosome representation and fitness function described in the section 4.2.1. Additional input is from the *Accumulator of unsuitable solutions* in which the former solutions, which does not meet demands for the maximal permitted delays, are stored

3. In the step: *Design of logical topology* the maximal data-flow delays are verified. The logical topology for the network must be designed for this purpose.

> The physical topology designed in the step No.2. is used as the input data together with the data set in the step one. If the delay is bigger than the maximal permitted delay, the physical topology is stored in the accumulator of the unsuitable solutions and the design continues again with the step No.2. in the next iteration. If the demanded network is not possible to design because of unrealistic demands (the data-flows are too big in comparison with the link capacity and therefore the delay is bigger than the maximal permitted delay) the algorithm could iterate forever. Thus, it is necessary to set the maximal number of permitted iterations. If the limit is exceeded, the network design is assumed as impossible since it is not possible to meets demanded attributes of the network.

The accumulator of the unsuitable solutions is possible to use during the physical topology design that can be influenced in different ways than only for a comparison with the unsuitable solutions from the previous iterations. This application of the accumulator is explained in the chapter 4.8.2.

4.6 Numerical results

The algorithm performance is the important attribute of the algorithm since it allows to judge if the algorithm is suitable for the task or not. Therefore series of tests for the medium sized network were done. The task was to design the physical topology for the network with the number of nodes $N \in \{10, 12, 14, 16, 18, 20\}$. The network should have different levels of the fault-tolerance in the different parts of the network. The main part of the network should be $1 - fault - tolerant$ and the more important part should be $2 - fault - tolerant$. Therefore, it is not possible to use the most common "safe" network topology - the ring topology; since the ring topology is only $1 - fault - tolerant$. Therefore, for the algorithm [1], [2] is necessary to design the whole topology as $2 - fault - tolerant$ and therefore the designed network is more expensive than the network designed by the above proposed algorithm that is able to design the network with different fault-tolerance levels.

All tests were done at $PC\ Pentium - 3GHz$, $512MB\ RAM$, $Windows - XP\ SP2$. The algorithm is implemented in $VS2005{-}C\#$. The stopping criterion for all tested algorithm was the number of iterations - 100. The number of chromosomes is also 100. The mutation probability is set to $p_m = 0.05$ and crossover probability $p_c = 0.20$. The cost matrix C has every element equals to 1 for purpose of easier comparison.

The Tab. 4.1 shows the time that is necessary for the design of the network with the same level of the fault-tolerance. We can see that the algorithm [1][2] is very fast. The k_F is the number of independent paths in the network $k_F = 3$ means that the whole network should be $2 - fault - tolerant$. We can see that the time necessary for the finding of the network topology is almost independent on the fault-tolerance

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

Table 4.1: Time consumption K-connectivity [1],[2]

level. Nevertheless, the algorithm is not very suitable for us since it is not able to ensure the real fault-tolerance (see chapter 2 **Reliability**).

N	μ	σ	min
10	$15,56$	0.05	15
12	18.96	0.06	18
14	22.7	0.06	21
16	25.55	0.09	24
18	28.85	0.09	27
20	32.66	0.1	31

Table 4.2: Costs K-connectivity [1],[2]

The Tab. 4.2 describes the acquisition costs for the network with the nodes N. The network should be $2-fault-tolerant$. The algorithm ([1],[2]) settings are the same as before. We can see that the algorithm is not able to find the minimum for every algorithm run (see columns min and μ - the average acquisition costs; the column min contains the minimal value found by the algorithm, the global minimum for the network with $N = 20$ is different than the minimum found by the algorithm). The statistical results are gained from the 100 runs of the algorithm. We can see that the algorithm settings for the network with the 20 nodes are not very good (see elements in the column σ, and inability to find the global minimum). The algorithm is possible to use if we want to get the results very fast and the fault-tolerance is not very important for us (see chapter 2 **Reliability**). These algorithms [1],[2] are not an option for us if we want to have the network which meets the demands for fault-tolerance. The network must be more expensive than it is necessary (the demanded fault-tolerance must be bigger than the necessary one - see chapter 2 **Reliability**). Therefore, the algorithm which uses the node degree as a fault-tolerance measure (see chapter 4.3) and is able to ensure that designed network is really fault-tolerant was proposed (the genetic algorithm uses the repairing functions described in the chapter 4.3).

Tab. 4.3 describes the time, which is necessary for the design of the network with

N	$T[msec.]$	$Costs$
10	482	15
12	759	18
14	1255	21
16	1689	24
18	2593	27
20	3228	30

Table 4.3: Time consumption - Same fault-tolerance

N nodes. The network is $2 - fault - tolerant$ (it is possible to interrupt two communication links and the network still allows the communication among nodes). The results were gained from 100 runs; the algorithm setting is the same as for the previous algorithm. We can see that this algorithm is slower than algorithm described in cite [1],[2].

Nevertheless, this algorithm was able to find the global minimum for every run. This is given by the size of the search space and by the conditions of the run of the algorithm. The allowable chromosome must meet demands for the $2 - fault - tolerance$. If the chromosome does not meet the demands for the $2 - fault - tolerance$, the repairing algorithm repairs it in order to meet the $2 - fault - tolerance$. Therefore, it is not possible that the network would contain more communications links than it is necessary for the fulfilling of the condition of the $2 - fault - tolerance$.

N	j	$T[sec.]$	μ	σ	min
10	4	2.86	13.41	0.05	13
12	5	4.41	16.42	0.08	15
14	6	6.49	19.38	0.08	18
16	7	9.39	22.11	0.09	20
18	8	13.42	25.27	0.09	23
20	9	18.62	28.63	0.11	26

Table 4.4: Fault-tolerance - different level

The Tab 4.4 contains results of the proposed algorithm, which is able to design the network with the different levels of the fault-tolerance in the different parts of the network. The biggest part of the network should be only $1 - fault - tolerant$ but the part of the network should be $2 - fault - tolerant$. The column j describes the number of the node-pairs that should have three independent paths - the core of the network (the interconnection of the node should be $2 - fault - tolerant$). The column T contains elements corresponding to the time that is necessary for the task solution. Column μ corresponds to the average value of the acquisition costs and column σ is the variance. Column min contains the minimal value of the acquisition costs that were found. The results were gained from the 100 runs of the algorithm and other settings are the same as for the algorithm described before.

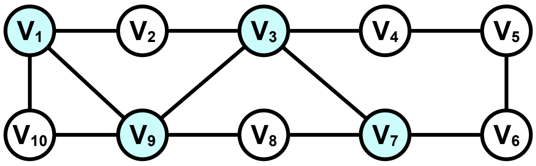

Figure 4.26: Fault-tolerant network-different level of fault tolerance

There is an example in the Fig. 4.26 of the network with 10 nodes in which every node-pair should be $1-fault-tolerant$ and minimaly 4 node-pairs should be $2-fault-tolerant$ (the first row in the Tab. 4.4). The connection between nodes V_1, V_9; V_9, V_3; V_3, V_7 and V_1, V_3 is $2-fault-tolerant$. Moreover, the connection between nodes V_9, V_7 and V_1, V_7 is $2-fault-tolerant$ as well. The network designed by the algorithm has similar structure for all rows in the Tab. 4.4. It is possible to see that even if two communication links in the network are disrupted the network core is still able to communicate, because there are three independent communication paths for every node pairs in the network core.

We can see that the algorithm is much slower than algorithm that uses the node degree as a fault-tolerance measure. It is caused by the application of the Ford-Fulkerson algorithm for the verification of the numbers of the independent paths that is very time demanded. On the other hand, the application of the Ford-Fulkerson algorithm allows to design the network with the different levels of the fault-tolerance in the different parts of the network. Thanks to that, it is possible to design the network that has exactly demanded fault-tolerance and therefore the acquisition costs are as small as possible. We can see it in the column μ if we compare it with corresponding columns in the Tab. 4.2 and Tab. 4.3. The average value of the acquisition costs of the network designed by the proposed algorithm for the different levels of the fault-tolerance is smaller than the minimal value of the acquisition costs of the network designed by other algorithms. This is caused by the abilities of the algorithms. The algorithm using the node degree as a fault-tolerance measure [1],[2], section 4.3 must design the network which meets demands for the maximal demanded fault-tolerance (see Fig. 4.27) and can do it very fast in comparison with the algorithm that is able to design the network with different levels of the fault-tolerance.

According to the results described in this section, one can concludes that the proposed algorithm is able to design the network with smaller acquisition costs than known algorithms [1], [2], [3] if there is a request for the different levels of the fault-tolerance in the different parts of the network but it needs longer time for this design.

It is better to use the algorithm that uses the repairing procedure described in the section 4.3 if we want to design the network that has the same level of the fault-tolerance in the whole network. The design is faster than with the algorithm for the

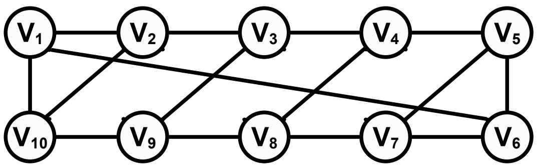

Figure 4.27: Fault-tolerant network, the same level of fault-tolerance

network with the different levels of the fault-tolerance and needs smaller number of the chromosomes for the global minimum of the acquisition costs finding.

4.6.1 Logical topology design

It is necessary to design the logical topology for the verification of the data-flow delay. The physical topology is one of the inputs to the logical topology design. We assume that data-flow is not possible to divide into more communication paths and data-flows are sent in data-packets of the constant length. Kleinrock interpolation (4.10) is used for the delay calculation. The genetic algorithm described in the section 4.4 **Logical topology design** is used for the design of the logical topology. The parameters of the algorithm are as follows: $p_c = 0.05$, $p_m = 0.20$. The stop criterion is the number of iterations. The algorithm stops after the 200 iterations and the number of chromosomes is 200 as well.

Series of tests were done for the verification of the time consumption and efficiency of the algorithm. The logical topology was designed for six medium sized network with the different numbers of nodes $N \in \{10, 12, 14, 16, 18, 20\}$. The physical topology of the networks meets demands for the $2 - fault - tolerance$ in the whole network (the networks are the results of the algorithm for the design of the network with the same fault-tolerance level in the whole network (see 4.3). It means that three independent communication paths exist for every data-flow and the logical topology should optimize the sum of the all delays of every logical path while any delay must not exceed the maximal permitted delay.

One can assume that the size of every data-flow is known before the logical topology design. Moreover, the data-amount of every node is not changed during the network operation and therefore the logical topology is not changed as well. The size of data-flows is possible to get from the knowledge of the control algorithm and data that are needed in every nodes for its correct functionality. For the purpose of the verification of the time demands of the logical topology design, it was assumed that every node in the network generates $10kb/second$ of the data and receives $10kb/second$ of data approximately, link capacity was $1000kb$. The size of data-flows was generated randomly in order to fulfill sent/received data-amount and nodes not necessarily must communicate with every other node (as in the real situation where not every node communicates with every other one).

N	$NofLink$	$Size[kB]$	$NofFlow$	$T[sec.]$	$T_flow[msec.]$
10	15	750	75	2.28	30.4
12	18	1464	122	4.16	34.1
14	21	2506	179	6.98	38.9
16	24	3728	233	9.77	41.93
18	27	5076	282	12.98	46.03
20	30	6620	331	16.55	50

Table 4.5: Time consumption - logical topology design

It is possible to see the time consumption of the algorithm for the logical topology design in the Tab. 4.5. N is the number of nodes $NofLink$ is the number of communication links in the network (In this case the whole network is the network core since node pairs should have the same number of the independent communication paths in the network); $NofFlows$ is the number of data-flows in the network. $Size$ is the length of one chromosome that represents a network logical topology ($Size = NofLink * NofFlow$). Size is in kB since the algorithm was written in the $C\#$ and the smallest space with which $C\#$ is able to work is one byte. T is the time necessary for the logical topology design. T_{flow} is the time necessary for the finding of the communication path for one data-flow. Data in the Tab. 4.5 was gained from a hundred runs of the algorithm for every number of nodes.

4.6.2 Simulation results

The end-to-end delay calculation applied during the logical topology design does not describe the network behaviour very precisely since it is used for the worse-case delay calculation. If the network meets the demands for the maximal permitted delay, we just know that the data are delivered until the maximal permitted time but we do not know the exact network behaviour. The delay calculation in the logical topology design expects that all data at every links are sent as one data-burst at every communication link and all other data at that communication link are put before the data for which we count the delay and this behaviour should be repeated at every communication link in the whole network. This behaviour is unlikely because it is not possible so that all data-flows are put before the data-flow for which we count the end-to-end delay at every communication link for every data-flow (see chapter 4.4.2). Moreover, it is not very probable that the data are sent as one burst. The data are split to the packets that are sent to their destination. Therefore, the end-to-end delay calculation used in the logical topology design does not describe the network behaviour but it is possible to use it as the upper boundary of the data-flow delay. Thus, it is useful to know the behaviour of the data-flows and common delay of the data in the network under demanded load as it is shown in the following paragraphs.

It is assumed that data are sent in packets of fixed size $Pack_{Len} = 50bytes$.

Every node uses an exponential distribution with the mean outcome of 0.005 for the packet generation ($Pack_{GEN} = 200$ packets per second). Packet routing is provided according to the routing table that was gained during the logical topology design. Data-flows routing is created for the data-flow delay calculation. It is possible to estimate the end-to-end delay thanks to knowledge of the data-load at every link, capacity of the link and characteristic attributes of the communication such as the packet size and the average packet outcome from the nodes. These variables allow to find all data-flows that effect the data-flow for which the end-to-end delay is estimated.

$$F_{\inf_L} = \sum_{j=0}^{n} f_{L_j} - \sum_{k=0}^{n} f_{L_k} \left(link_k - 1\right) \tag{4.21}$$

Where the L is the path index; $\sum_{j=0}^{n} f_{L_j}$ is the sum of all data-flows going through the path L and j is the index of data-flow going through the path L. $\sum_{k=0}^{n} f_{L_k} \left(link_k - 1\right)$ is a sum of all data-flows going through the communication path L and that does not have any influence on the packet for which the end-to-end delay is counted. $Link_k$ is the number of links that are at the communication path for the data-flow k and data-flow for which the end-to-end delay is counted. If the data-flow k causes a delay of the data-flow for which we count the end-to-end delay at one communication link, it cannot cause the other delay at another communication link. The delay caused at the first link is long enough that data-flow k can leave other communication link before the data-flow, for which we count the delay, enters to next link. This assumption is valid if the communication is packet switching.

Then the average number of packet that can cause the delay at the communication path L is:

$$Pack_L = \frac{F_{\inf_L}}{Pack_{Len}}, \tag{4.22}$$

where $PackLen$ is the size of the packet.

It is assumed that the data in the network are sent in packets and communication is losses free. It is not possible so that the number of packets grows steadily at every link and incoming queue of nodes. The number of packets at the communication link and input queue must be stable from the long-term point of view during the steady communication. Therefore, it is possible to expect that if node generates the packet, all other nodes generated by this node leaves the communication link and input queue (it is not valid for packets that are retransmitted by this node). Then the average number of packets that are able to delay packet for which we count the end-to-end delay is:

$$Pack_D = \frac{Pack_L}{Pack_{GEN}} = \frac{Pack_L}{200}. \tag{4.23}$$

Where $Pack_{GEN}$ is the average packet outcome from the node per second (in this case 200).

Then the average end-to-end delay of the packet including the delay caused by the other load of the network is:

$$ETE_L = (Pack_D + link)\, T_p. \tag{4.24}$$

Where $link$ is the number of communication links on the communication path L and T_p is the time that needs a packet for passing through one communication link and input queue. T_p is possible to count according to (4.10).

If there is any other load at the network, then end-to-end delay is:

$$ETE_L = T_p\, link. \tag{4.25}$$

The estimation of the end-to-end delay is necessary to verify. The OPNET software is possible to use for this verification. The OPNET software is an environment that allows fast developing and prototyping of the networks and communication protocols. OPNET contains the basic models of the communication links and the network technologies from which it is possible to create own network. Moreover, it is possible to modify the existing model or create own model. We use this attribute for creation of our own model of the communication network (the frequency of packet generations, link capacity, packet size is the same as for the designed network).

4.6.2.1 OPNET model

The OPNET model comprises the nodes and the communication links. Every node cannot only receive and transmit data packets but it is also able to route the packets from other nodes. Therefore, it is possible to split every node into two parts: the data-processor parts and the switch.

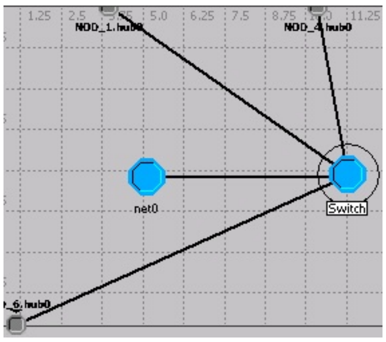

Figure 4.28: OPNET node

The switch model routes the data-packets according to the routing table that was created by the algorithm for the logical topology design. The routing table is static during the whole operational time of the network and it was transformed into the OPNET environment as a header file.

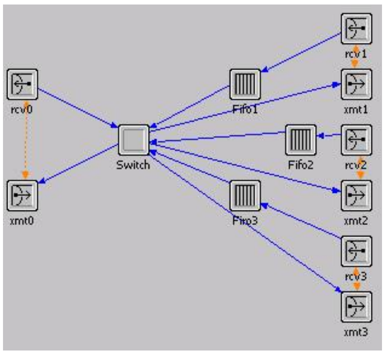

Figure 4.29: OPNET - switch

It is possible to see the OPNET switch model in the Fig. 4.29. We can see that it is possible to divide model into the “Node part” (left) and the “Network part” (right). The “Node part” provides a connection of the switch and the actual node. The “Network part” interconnects parts of the network. It is possible to connect three communication links into the switch. The switch works in full-duplex mode (it is possible to transmit packets and receive them at the same time). At the incoming link there is a FIFO queue that ensures that no packet is lost if there is some unreceived packet. The FIFO queue produces the delay that affects the incoming packets but there is any other delay between node and the switch (it means that packets between the node and the switch at the “node part” are sent without any additional delay).

The model of the Node contains packet source src that generates packets of 50 bytes length at frequency $Pack_{GEN} = 200$ packets per second. These packets go to the $proc$ unit that fills the destination address into the packet according to the data that were gained from the algorithm for the logical topology design. Then packets continue to the switch that routes packets into the communication link according to the routing table. The packet frequency and their direction correspond to the matrix of data-flows that was used during the logical topology design. The incoming packets into the node are destroyed and their end-to-end delay is stored in the network statistics.

The every part of the OPNET model works as the state automaton. Automaton reacts at every state to external events (incoming/outgoing packet) or to internal events (timers, stochastic event, ...).

The process model of the node is depicted in the Fig. 4.31. The process $init$ is an initialization of the state automaton. The process $proc$ node handles with all events that occur during operational time of the node. There are two events: SRC_ARRVL and RCV_ARRVL. Both of them are connected with the packet arriving to the node.

The event SRC_ARRVL occurs when a packet from the source src arrives to

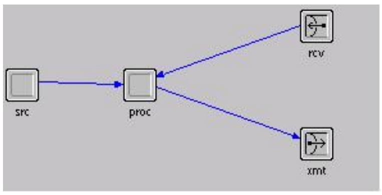

Figure 4.30: OPNET - node

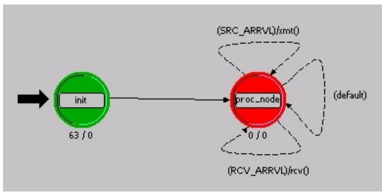

Figure 4.31: OPNET - node - process model

the nodes. The procedure $xmt()$ that is launched by the event SRC_ARRVL fills the address of the destination node and the source node into the packet according to the matrix of data-flows and then packet is sent to the $switch$.

Event RCV_ARRVL occurs if the packet from the network arrives to the node. The procedure $rcv()$ is launched and the end-to-end delay of the arrived packet is stored into the statistics. After then, the arrived packet is destroyed.

The model of the switch works in the similar manner but there is only one event and procedure, since switch only retransmits packets from the arriving communication link into the outgoing communication link according to the routing table.

OPNET modeller allows simultaneous run of the several models at the same time, thanks to that it is possible to create the whole network from the nodes connected together via communication links.

It is possible to see an example of the network created in the OPNET Modeller in the Fig. 4.32. The network comprises 8 nodes and 15 node pair have three independent communication paths and create the network core (in this case every node with 3 communication links belongs to the core). Other nodes have two independent communication links. The algorithm for the logical topology design created the logical topology and its parameters were transformed into the OPNET environment. There are 52 data-flows in the network. Statistical behaviour of the network was collect thanks to the OPNET.

It is possible to see the example of the statistic of the End-to-end delay in the Fig. 4.33. One can compare the statistics gained by the OPNET software (light blue line) and the estimation of the network behaviour gained by our algorithm (black, blue, green and red colour).

The light-blue colour is a statistic gained by the OPNET software during simu-

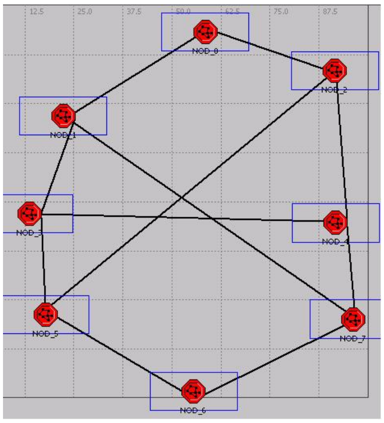

Figure 4.32: OPNET network

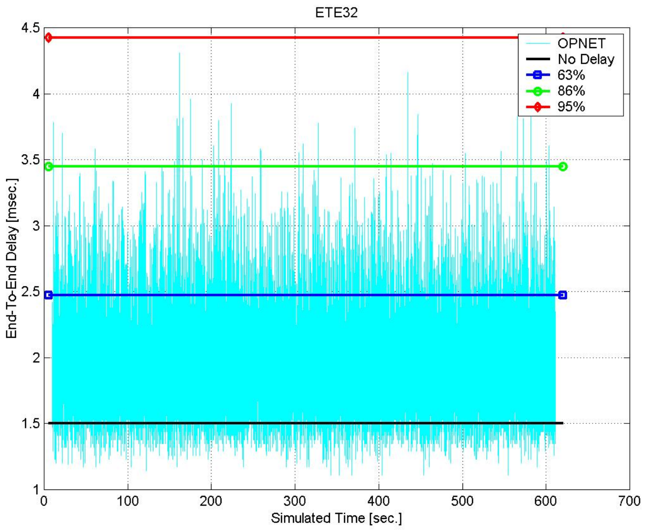

Figure 4.33: End-to-End delay OPNET

lations. The black line corresponds to the end-to-end delay estimated by model of common behaviour for the situation where there is no other data-flow in the network. The blue line with the square ends corresponds to the end-to-end delay estimated model of common behaviour for situation if there are other data-flows in the network according to the data-flow matrix. Under this line there should be 63% of the end-to-end delays of the statistics gained by the OPNET software. Under the green line with the circle ends, there should be 86% of the end-to-end delays of the statistics gained by the OPNET software. Finally, under the red line with the diamond ends, there should be 95% of the statistics gained by the OPNET software.

$Model$	63%	86%	95%
$Opnet$	93%	98%	99%

Table 4.6: Comparison of End-to-end delay of the OPNET and proposed model

Tab. 4.6 shows the comparison of the statistics of the end-to-end delay of dataflows. The results were gained by the proposed model and OPNET. The number of packets that should reach the destination sooner than estimated time are in the row $Model$. The real numbers of packet that really reaches destination before estimated time are in the row OPNET.

We can see that the estimation of the end-to-end delay is more pessimistic than it is the real network end-to-end delay. Nevertheless, the estimation allows to predict behaviour of the real network before the actual network is built since the estimation can set the lower bound of the end-to-end delay and say the common end-to-end delay. The end-to-end delay counted during the logical topology design is so pessimistic that it gives the constrain for the worst-case scenario but this situation can happen only at some communication link and not in the whole network. Thus, the estimation described in this chapter says much more about the real network behaviour.

4.7 Time complexity

The genetic algorithm is used for the network topology design since there is no analytical algorithm that is able to design medium sized network. The Ford-Fulkerson algorithm allows verification and finding of the independent communication paths. Then it is possible to describe the running time of the algorithm by the following equations.

4.7.1 Time complexity - physical topology design

$$\begin{aligned}\Theta_P = N_{ch}\left(\frac{N\,(N-1)}{2}\,(2+|E|\,n_{\max})\right)+\\ +N_{ch}\left[(2p_c+p_m)\left(|E|\,n_{\max}\frac{N\,(N-1)}{2}+2\right)+4\right]N_p \qquad (4.26)\end{aligned}$$

The first part belongs to the initialization of the genetic algorithm and the second part of the equation belongs to the iterations of the genetic algorithm for the physical topology design. N_{ch} is the number of chromosomes used for the physical topology design; N is the number of nodes in the network $|E|$ is the number of communication links in the network; n_{max} is the maximal number of the independent communication paths in the network for some node-pair; p_c and p_m are the crossover probability and the mutation probability respectively; N_p is the number of genetic algorithm iterations.

The part that is not possible to omit it is the complexity of the Ford-Fulkerson algorithm since it is necessary to use it for every node-pair that should have independent communication paths. If the network meets certain conditions, than the complexity of the Ford-Fulkerson algorithm is:

$$\Theta_{FF} = |E|\, n, \tag{4.27}$$

where n is the maximal number of the independent paths for the node-pair in this case. Ford-Fulkerson algorithm finds the maximal permissible flow in the network. It is possible to transform the search of the maximal flow into the search of the number of the independent paths if we set capacity of every communication link to 1. Then, the maximal flow, which was found, is the number of the independent paths between the source and the sink node as well. If the bread-first search algorithm is used for finding of augmenting paths between the source and the receiver and the capacity of the communication links is integer, then the running time of the algorithm is (4.27) [31]. These conditions are met automatically in this case (the bread-first search algorithm is used and capacity of the communication link is 1). The time, which is necessary for chromosome creation, corresponds to the length of the chromosome and it is in this case $\frac{N(N-1)}{2}$. The same complexity has the evaluation of the chromosome as well. All these operations (the chromosome creation, the verification of the number of the independent paths and the evaluation) must be done N_{ch} times during the algorithm initialization.

Fewer chromosomes are created during the iterations of the genetic algorithm than during the creation of the initial population. The number of the new chromosomes is $(2p_c + p_m)N_{ch}$ - there are two offspring after the crossover operations. It is necessary to verify the number of independent paths for every node pair at the every newly created chromosome. Off course, that it is necessary to create chromosome firstly and then evaluate it as well (element 2 in the equation). During every algorithm iteration is necessary to choose and copy chromosomes into the new population (element 4 in the equation). It is necessary to repeat all these operations N_p times (the number of the iterations of the genetic algorithm).

The genetic algorithm is used for the design of the fault-tolerant network with the different levels of the fault-tolerance. Therefore, it is possible to assume that there are 2 or more independent paths in the network for the most of the node pairs. Then the maximal number of the independent paths is:

$$n_{\max} \in \langle 2, N-1 \rangle . \tag{4.28}$$

2 independent paths are for the ring topology and $N-1$ path are for the fully connected network (every node is connected with all other nodes).

Then the complexity of the Ford-Fulkerson algorithm according to the number of the communication links is:

$$\Theta_{FF} = \left\langle 2N, \frac{N\left(N-1\right)^2}{2} \right\rangle . \tag{4.29}$$

Then the verification of the number of independent paths in the whole network is:

$$\Theta_{Ch} \in \left\langle N^2\left(N-1\right), \frac{N\left(N-1\right)^2}{2} \frac{N\left(N-1\right)}{2} \right\rangle . \tag{4.30}$$

It is possible to approximate the time, that is necessary for the verification of the number of the independent paths in the whole network, as:

$$\Theta_{Ch} \in \left(N^3, N^5\right) . \tag{4.31}$$

Then the time necessary for the run of our algorithm is possible to approximate as:

$$\Theta_P = N_{ch}\left(N\left(N-1\right) + \Theta_{Ch}\right) + N_{ch}\left[\left(2p_c + p_m\right)\left(\Theta_{Ch} + 2\right) + 4\right] N_p . \tag{4.32}$$

Where the biggest elements always belongs to Θ_P. Then the time complexity for the run of the proposed algorithm is:

$$\Theta_P \in \left(N^3, N^5\right) . \tag{4.33}$$

4.7.2 Time complexity - logical topology design

The logical topology design is used for the verification of the size of the maximal permitted delay. It is necessary to know the maximal delay of the data-flows in the network since some control algorithm can be very sensitive for lack of the data or its late delivery. The logical topology design needs certain time for the optimization of the communication paths. Therefore, it is good to know the time complexity of the logical topology design.

$$\begin{aligned}\Theta_L = {} & \left(\left|E\right| n_{\max} N_F + N_{LCh} N_F 2 \left|E\right| N_F\right) + \\ & + \left(N_{LCh}\left[4 + \left(\left(2p_{Lc} + p_{Lm}\right)\left(1 + 2\left|E\right| N_F\right)\right)\right] N_L\right) .\end{aligned} \tag{4.34}$$

Where the first part of the equation belongs to the genetic algorithm initialization and the second part belongs to the iterations of the genetic algorithm. N_F is the number of data-flows in the network; N_{LCh} is the number of chromosome used by the genetic algorithm; p_{Lc} and p_{Lm} is the probability of the crossover and mutation respectively and N_L is the number of iterations of the genetic algorithm.

It is necessary to find all possible communication paths for all data-flows at the start of the logical topology design; complexity of this task corresponds to: $|E|\, n_{\max} N_F$. After finding of all possible communication paths, the chromosomes must be created and evaluated:

$$N_{LCh} N_F 2\, |E|\, N_F. \tag{4.35}$$

The initial population of the chromosomes is created and evaluated during these tasks and it is possible to continue at iterations of the genetic algorithm.

The new chromosomes are created during the every iteration of the genetic algorithm. The number of created chromosomes corresponds to the probability of the crossover and mutation $(2p_{Lc} + p_{Lm})$. It is necessary to evaluate the newly created chromosomes $(2\,|E|\, N_F)$. After that all chromosomes at the new iteration were created, it is necessary to choose the chromosome for the next generation and create the new chromosome population (element 4 in the equation). All these operations must be done during every iteration (N_L) of the genetic algorithm.

If it is possible assume that every node communicates with all others nodes in the network in the worst-case, then we can expect that:

$$N_F \approx N(N-1) \approx N^2. \tag{4.36}$$

Then,

$$\begin{aligned}\Theta_L = {} & (|E|\, n_{\max} N_F + N_{LCh} N_F 2\,|E|\, N_F) + \\ & + (N_{LCh}\,[4 + ((2p_{Lc} + p_{Lm})\,(1 + 2\,|E|\, N_F))]\, N_L)\,.\end{aligned} \tag{4.37}$$

The estimation (4.35) is for the worst-case but it is hardly possible to assume that every data-flow goes through the all communication links in the network. This is actually not possible since if the data-flow passes all the communication links and if the network topology is fault-tolerant. The data-flow should pass through the node which transmits it in order to pass all edges and it is not definitely very good logical topology at all. In real scenario, there is an effort to design the logical topology in way that the data-flows must pass as less communication links as possible and number of the communication links that is used by the data-flow is very small. Then the estimation of the time complexity corresponds to:

$$\Theta_L \approx N^4. \tag{4.38}$$

Then the time complexity of the whole algorithm for the network topology design is:

$$\Theta = (\Theta_P + \Theta_L)\, N_A. \tag{4.39}$$

Where N_A is the number of iterations of the whole algorithm (It occurs if the verification of the maximal permitted delay of the data-flows was unsuccessful and it is necessary to design the physical topology again with the knowledge that the already designed network is impermissible).

One can approximate the time complexity of the network design as:

$$\Theta \in \left(N^4, N^5\right). \tag{4.40}$$

The time complexity is quite high but allows to find the network topology design in the reasonable time (unfortunately the algorithm is not able to ensure so that the solution will be the optimal one - it is general attribute of the genetic algorithm).

4.8 Algorithm extension

The described algorithm is able to design the fault-tolerant network with the different levels of the fault-tolerance in the different parts of the network. This network can have the exact demanded fault-tolerance and can be less expensive than the network designed by the other algorithms [2],[6] but still may not meet the real life demands since it does not consider the limitation of the real control systems such as Programmable Logical Controller.

Every real communication node has a constrained number of the communication ports and the algorithm for the network design can produce a such network topology which needs more communication ports at some node than it is available at the part of the control system. The common solution is the increase of the communication ports of the nodes but at some situation, it is not possible. Therefore, it is necessary to introduce this limitation into the algorithm. Fortunately, the genetic algorithm allows easy implementation of several constraints thanks to the penalty function.

It is necessary to know the number of the communication ports of every node for the application of the number of communication ports constrain. This information is stored in the variable K. Then we just change the evaluative algorithm for the physical topology design. We change the penalty function (4.6) in following way:

$$Pen = \left(kN^2 + rN^3\right) c_{\max}. \tag{4.41}$$

Where k is the number of nodes that should have bigger degree according to the designed network than it is the real number of the communication ports. It is possible to see that it is more important to meet demands for the fault-tolerance than the conditions for the number of the communication ports. The application of the condition has sense only if the demanded number of the independent communication paths is smaller or equal to the number of communication ports at the transmitting/receiving nodes. It can happen that there is not possible to design the topology network without an exceeding of the number of the communication ports available at the node. Then, the network designer must decide whether it is possible to increase the number of the communication ports at the node with small number of the communication ports or change the requirements put on the network.

4.8.1 Different physical layers

There could be a limitation of the applied physical layer as well. Then, it is necessary to change the algorithm for the design of the physical topology. The number of the input parameters increases since it is necessary to add the matrix of the acquisition costs for every kind of the physical layer. The evaluation is also necessary to change since it must correspond to a possible application of more physical layers.

$$C_P = (C_1 \vee C_2 \vee ... \vee C_n)\,\&P. \tag{4.42}$$

The matrix C_P is the cost matrix for the designed network. Matrices $C_1, ..., C_n$ are matrices of the acquisition costs for the different physical layers. The matrix of the acquisition costs for the cost calculation is chosen correspondingly to the physical layer used at the communication link. The operator & works in the same way as in the equation (4.3). The elements of the matrices $C_1, ..., C_n$ are chosen correspondingly to the kind of the physical layer. The penalty function that is applied if the network does not meet the requests for the fault-tolerance or violate the limitation of the node degree is the same as (4.41).

It is necessary to change the used chromosome representation if there is possibility to use more than only one physical layer. The chromosome representation is based on the adjacency matrix. It is possible to use the adjacency matrix for the network representation but it is necessary to change it since the adjacency matrix describes the connection between nodes but not the kind of the connection. Therefore, if one can assume that the elements of the “adjacency” matrix can be not only 0 and 1 but also different numbers, then the “adjacency” matrix can describe the kind of nodes interconnection as well. It is possible to see in the Fig. 4.34. It is possible to see that there is no change in transformation of the “adjacency” matrix into the chromosome describing the network configuration. The element 2 in the “adjacency” matrix means that the fibre cable connects the node-pair; element 1 in the “adjacency” matrix means that the metallic cable connects the node pair; element 0 means that there is no direct connection of the node-pair.

The network, described by the chromosome in Fig. 4.34, is depicted in the Fig. 4.35. The chromosome coding has a direct influence on the size of the memory that must be allocated for a chromosome. If the network uses only one physical layer, it is possible to use only bits for the chromosome coding. Thanks to that, it is possible to save the memory. If the network with the different physical layers is designed, it is necessary to use the chromosome coding that must be able to describe all possible physical layers in the network. A byte is used for one gene coding. It allows to use 128 different physical layers and it is definitely more than one can need.

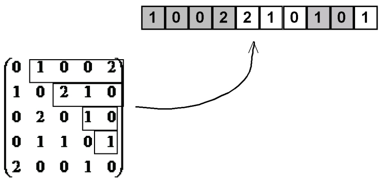

Figure 4.34: Chromosome representation - more physical layers

The memory amount necessary for one chromosome storing should be eight times bigger than memory necessary for one chromosome of the network with the only one physical layer. Nevertheless, in this case, it is not possible to use so straightforward estimation because of the huge influence of the algorithm implementation. The algorithm is implemented in C# and the .NET framework allocates a byte for every binary variable. Then there is not possible to find any change in the size of the allocated memory if the network with the different physical layers is designed in comparison with the design of the network with only one physical layer.

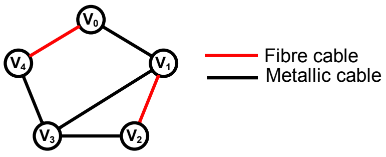

Figure 4.35: Network - more physical layer

If C_F and C_M are matrices of the acquisition costs for the physical layer from the fibre cables or the metallic cable respectively. Then the fitness of the network depicted in the Fig. 4.35 without penalty function looks like:

$$c_{ost} = c_{M0,1} + c_{M1,3} + c_{M2,3} + c_{M3,4} + c_{F1,2} + c_{F0,4}. \tag{4.43}$$

The fitness of the chromosome is the same as the acquisition costs of the network that could be created according to the chromosome. If the chromosome representation is changed it is necessary to consider modification of the genetic operators as well. The crossover operator uses two chromosomes for creation of their offspring. It chooses parts of the original chromosomes and crosses them in order to create their offspring. The crossover operator does not put into the offspring any element which is not included in parent chromosomes. Therefore, common one point crossover can deal with the different physical layers and it is not necessary to modified it. On the other hand, “one bit mutation” only overlaps values of bit. Therefore, it is not possible to use it if we design the network from more physical layers since the common “one bit mutation” is not able to accommodate them. Therefore, it is necessary to modify the mutation operator to accommodate more than only one physical layer. The mutation operator must still work on the basis of randomness and it works according to the following algorithm:

1. Randomly choose the chromosome that is going to be mutated.
2. Randomly choose the gene that is going to be mutated.
3. Randomly change value of chosen gene to the element from the set that contains all elements describing different physical layers and no connection.
4. Repeat steps No. 1..3 until you gain the demanded number of mutated chromosomes.

Described chromosome representation, fitness function and genetic operators allow to design the network that can contain different physical layers and thanks to that more complex physical topology of the network.

4.8.2 Prohibited topologies

There can be special demands for the physical topology if the network is designed for the application in the industry. There can be request that the network must not have certain configuration of the node interconnections because of the limitation arose from the limitation of the industrial environment e.g. some nodes cannot be connected directly and so on. This limitation must be applied in the design of the physical topology. It is possible to do it thanks to the structure of the proposed algorithm that allows giving information about the unsuitable topologies directly to the algorithm for the physical topology design (see Fig. 4.36).

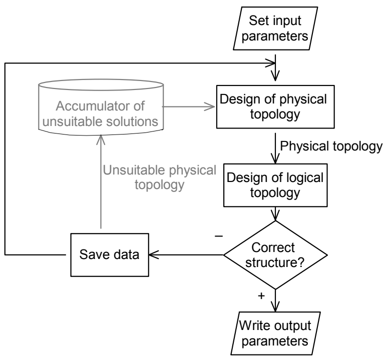
Set input parameters
Accumulator of unsuitable solutions
Design of physical topology
Physical topology
Unsuitable physical topology
Design of logical topology
Save data
−
Correct structure?
+
Write output parameters

Figure 4.36: Algorithm - network design

We can see the algorithm part that allows using information about the inappropriate topologies during the network design. When the whole algorithm is initialized, not only the matrices of data-flows and acquisition costs are loaded but also all unsuitable topologies are stored into the accumulator of the unsuitable topologies. The algorithm can use knowledge of the unwanted topologies thanks to the evaluative function and its penalty part that is as follows:

$$Pen = \left(kN^2 + (r+u)\,N^3\right) c_{\max}. \tag{4.44}$$

Where k is the number of the nodes at which the designed degree is bigger than the real node degree; r is the number of the node-pairs that do not have the demanded number of independent communication paths and u is a variable that is set according to the accumulator of the unsuitable topologies in the following manner: $u = 1$ if the network described by the chromosome is already in the accumulator of the unsuitable topologies; $u = 0$ if the chromosome is not in the accumulator of unsuitable topologies.

The structure of the algorithm and penalty function (4.44) allows designing the network exactly according to wishes of the application engineers. It allows them to choose the characteristics of the network exactly according to the requirements of the controlled technology. The application engineer can choose not only the level of the fault-tolerance in the different parts of the network but also he can force the algorithm to design the network according to the possibilities of the environment in which the network will be used.

4.8.3 Network expansion and reduction

In the real world situation, there are often demands for a modification of the existing network rather than the demands for creating of the brand new networks. This necessity can occur because of changes in the controlled technology or in the control systems. The changes in the controlled technology can produce changes not only in the amounts of the data-flows but also in the number of parts of the control system. There have been published papers that deal with the problem of the computer network enhancement [3],[23]. These papers assumed that the number of parts of the network (PCs) is still the same and only the size of data-flows is changed. It means that these algorithms should redesign the backbone of the computer network. The main task of these algorithms was to design the new routing of the data-flows in the network. If the network is not able to transfer the demanded amount of data-flows, these algorithms are able to add communication links into the network in order to network capacity increase. These algorithms deal only with the expansion of the network and they do not consider that sometimes not only the network expansion is necessary but the reduction is needed as well. The necessity of the network reduction can occur since the number of the parts of the network is reduced. It means that if it is possible to exclude the part of the network systems, these algorithms design such network in which all communication parts of the original network still participate in the data exchange in the network. It means, that the communication part of the possibly excluded system is not possible to use somewhere else and it brings additional costs to the owner of the communication network. Moreover, these algorithms did not count the costs that are connected with a removal of a communication infrastructure. These algorithms do not apply the limitation of the number of the communication ports at the devices and do not allow the application of a priori information about unsuitable topologies that are not possible to use for some reason. It is not possible to use the unsuitable physical topologies mostly because of the physical limitation of the environment in which the network will be applied.

The above mentioned characteristics of the already existing algorithms give the reason to propose the algorithm for the network expansion/reduction that is able to design the network with the requested changes in the number of nodes and data-flows. Moreover, the algorithm ensures that the network has the demanded fault-tolerance for every node-pair and allows to use a priori information about the unsuitable topologies of the network that is not possible to create.

The algorithm for the design of the network with the different levels of the fault-tolerance is used as a core of the algorithm for the network reduction/expansion. The algorithm can be used in two possible situations. Firstly, it is the reduction of the network and another possibility is the expansion of the network. Both operations use different technique for the network design and they will be explained in the different paragraphs. There is also possible that it is necessary to reduce and enlarge the network at the same time. This task is a combination of the both mentioned above.

It is necessary to add some variables to the algorithm since it needs additional information for the network reduction/expansion. One of variables is the matrix of reduction costs C_R. This matrix is necessary to use if the network is reduced as well as if the network is expanded. If the network is expanded and there is no limitation of the number of the communication ports at nodes, then the less expensive change of the network is an addition of the communication links to the network. If there is a limitation of the communication ports at the nodes, the removal of some communication links and the addition of the new communication links can be cheaper than only an addition of other communication link. Another variable, which is necessary to add to the algorithm, is a list of nodes that should be removed. The list R_{ed} contains all nodes that should be removed.

The algorithm for the network expansion/reduction does not create the whole initial population at random since we have the information about the topology that should be similar as the optimal topology of the expanded network. This similar topology is an original topology that should be changed. Therefore, it is possible to use this knowledge for the design of the 75% of the initial population and only 25% is created at random.

4.8.3.1 Network expansion

The number of the nodes that are added to the network during the network expansion is possible to find thanks to the increased size of the matrices of the fault-tolerances, acquisition costs etc. The initial population of the genetic algorithm is created in the following way:

1. Create chromosome that is able to code the whole network including the added nodes and describes the original network.

2. For the chromosome part, that describes the original network: Randomly decide if the gene will be changed. If yes, randomly choose if there will be a connection of the nodes and material of the connection or if the existing connection will be removed. The connections can be added only until the limitation of the communication ports is not reached.

3. For the chromosome part, that describes the new part of the network: Randomly decide for every gene if there will be connection and type of the connection that is described by the gene. It is possible to add connections only until the limitation of the number of communication ports is not reached.

4. Repeat steps No. 1..3 until the 75% of the initial population is created.

The rest of the initial population is created at random without an influence of the original network. The limitation of the number of the communication ports is the only limitation used during the random generation of the rest of the initial population. It means that it is not possible to add the communication link to the node that already has no communication port available.

It is necessary to evaluate the initial population after its creation. A mask that corresponds to the chromosome describing the actual network without any added communication link is created for this purpose. The chromosome is compared with the mask and only differences between the mask and chromosomes are evaluated.

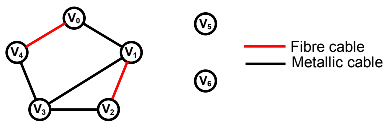

Figure 4.37: Network expansion - original

It is possible to see the original topology of the network in the Fig. 4.37. Two nodes V_5 and V_6 should be added into the network. The structure of the original network and the “adjacency matrix” with added nodes is depicted in the Fig. 4.38.

The yellow boxes in the Fig. 4.38 shows the part that was added to the “adjacency” matrix and the chromosome as well. The chromosome depicted in the Fig. 4.38 is also the mask applied during the chromosome evaluation.

It is possible to see the original network topology and the topology that was created during the chromosome creation of the initial population of the genetic algorithm

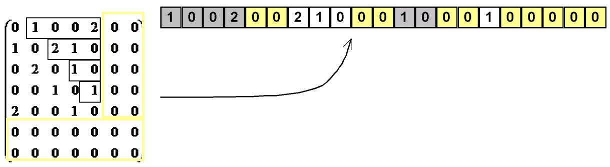

Figure 4.38: Network expansion - original

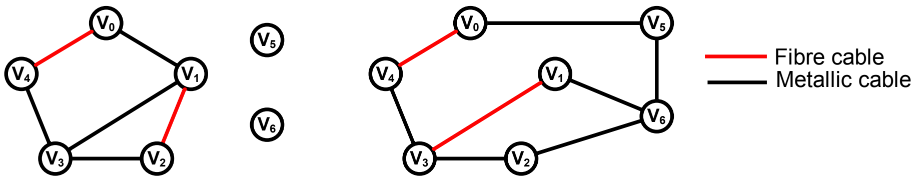

Figure 4.39: Network enhancement-initial population

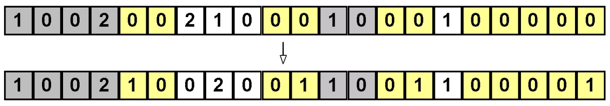

Figure 4.40: Network expansion - initial chromosome

in the Fig. 4.39. We can see that there have been added not only the new communication links but also the original topology has been changed as well. Moreover, the physical layers used in the original network were changed in the new network as well.

We can see the chromosome describing the original matrix and the chromosome that was created for the initial population of the genetic algorithm in the Fig. 4.40. The original chromosome is in the upper part of the figure. The new chromosome is in the bottom part of the figure. The original chromosome is the mask for the evaluation of the new chromosome as well. Then the fitness and acquisition costs are:

$$c_{ost} = c_{M0,5} + c_{M1,6} + c_{M2,6} + c_{M5,6} + c_{F1,3} + (c_{RM0,1} + c_{RF1,2} + c_{RM1,3}). \quad (4.45)$$

Where elements $c_{Mi,j}$ are the acquisition costs for a creation of the connection between nodes i and j. This connection is from the metallic cable. $c_{Fi,j}$ is the acquisitions costs for a creation of the communication link between nodes i and j with the help of the fibre cable. On the other hand, elements c_{RM} and c_{RF} are the costs that are necessary to pay if we remove the fibre cable and metallic cable between nodes i and j. The elements in the brackets correspond to the costs that are necessary to pay for the removal of the communication links. If the communication cables are left in the network, these elements are equal to zero. We can see that the fitness function evaluates only the changes in the network. Moreover, the same penalty function, as for the design with the help of the knowledge of prohibited topology, is used (see chapter 4.8.2). The application of this penalty function is necessary since it is possible that the created chromosome describes the unsuitable topology that is not possible to use for the network construction.

The genetic operators used for the network enhancement are the same as those for the design of the physical topology with the different physical layers. Thanks to that, it is possible to design the network enhancement with the application of the different physical layers. The fitness function is the same as for the chromosomes created for the initial population. It allows to design the network with the application of a priori information. The network can be designed from the different physical layers and does not violate the limitation of the number of the communications ports. The stop rule of the genetic algorithm is the number of the algorithm iterations. The number of iterations is 100.

The modification of the fitness function, genetic operators and creation of the initial population allows to design the network enhancement not only for the same number of the communication nodes but for bigger number of the nodes as well. This and possibility of the application of a priori information are the main advantage in comparison with the method described in [3], [23], [24]. This network expansion allows to redesign the original network in order to meet demands of the new control system that is necessary to use if the controlled system is changed.

4.8.3.2 Network reduction

The algorithm for the network reduction is used if it is necessary to remove parts of the network. This can occurs when the control system is changed. The algorithm should be able to use the knowledge of the inappropriate solutions that can be possible to gain from the owner of the network. Moreover, the algorithm must incorporate the limitations of the number of communication ports and the node-pairs should have the demanded fault-tolerance. It is necessary to know the list of nodes R_{ed} that should be removed from the network for the design of such network. If these nodes are removed, then the communication paths that go via them are interrupted and therefore it is necessary to redesign the network.

It is possible to use a priori information for the creation of the initial population. 75% of the initial population is created with the help of a priori information and 25% is created at random. The chromosome has length that satisfies the description of the original network. The chromosome that is created with the help of a priori information uses following procedure for its initialization:

1. Create the chromosome that describes the original topology.
2. For every gene in chromosome: Verify whether gene corresponds to the connection to the node that should be removed. If yes, set the gene to 0 and continue with other genes. If no, continue to step No. 3.
3. Randomly decide if the gene will be changed. If yes, randomly choose change

of the gene (connection is possible to add only up to the number of communication ports of certain type available at the node).

4. Repeat steps 1..3 until 75% of initial population is created.

It is necessary to evaluate the chromosomes after their creation. The same fitness function as for the network expansion is used - only the penalty function is modified to:

$$Pen = \left(kN^2 + (r + u + e_r)\,N^3\right)c_{\max}. \tag{4.46}$$

Where k is the number of nodes at which the designed node degree is bigger than the real node degree; r is the number the node-pairs that does not have demanded number of independent communication paths and u is a variable that is set according to the accumulator of the unsuitable topologies in the following manner. $u = 1$ if the network described by the chromosome is already in the accumulator of the unsuitable topologies; $u = 0$ if the chromosome is not in the accumulator of unsuitable topologies. e_r is the variable that corresponds to the number of the communication links connected to the nodes that should be removed and have not been removed. The same fitness function with the penalty function is also used during the iterations of the genetic algorithm. Therefore, there are elements that cannot be used during the creation of the initial population such as k and e_r. The chromosome in the initial population does not describe more connections to some nodes than it is the number of the communication ports and there cannot be a connection to the nodes that should be removed. Both attributes are checked during the chromosomes creation.

The genetic operators used for the network reduction are the same as those for the design of the physical topology with the different physical layers. Thanks to that, it is possible to design the network reduction with the application of the different physical layers. The fitness function is the same as described above. It allows to design the network with the application of a priori information. The network can be designed from the different physical layers and does not violate the limitation of the number of the communications ports. The stop rule of the genetic algorithm is the number of the algorithm iterations. The number of the algorithm iterations is 100.

4.8.3.3 Network reduction and expansion

It is possible that it is necessary to reduce and expand the network at the same time. The fitness function, genetic operators and stop rules are the same as for the network reduction. The generation of the chromosome for the initial population is a combination of the algorithms for the network reduction and expansion. The algorithm is as follows:

1. Create chromosome, which is able to represent the original network with the added nodes. The chromosome describes the original topology.

2. For the chromosome part, that describes the original network, for every gene:
   (a) If the gene describes the connection of the node that should be removed: set the gene to 0 and continue with other gene.
   (b) Randomly decide if the gene is going to be changed. If yes, randomly choose whether there will be a connection of the nodes and material of the connection or if the existing connection will be removed. The connections can be added only until the limit of the number of the communication ports is reached.
3. For the chromosome part, that describes the new part of the network for every gene:
   (a) If the gene describes the connection to the node that should be removed, set the gene to 0.
   (b) For other gene randomly decide whether there will be connection and type of the connection that is described by the gene. It is possible to add connections only until the limitation of the number of the communication ports is reached.
4. Repeat steps 1..3 until the 75% of the initial population is created.

75% of the chromosomes is created according to the above described algorithm, other 25% is created at random. The fitness function for the initial population is the same as for the network reduction.

It is possible to redesign the whole network thanks to the algorithm described above. It is possible to change not only the node interconnection but also it is possible to add and remove nodes from the network at the same time. It allows to reduce possible costs for the network redesign since the algorithm knows about all nodes that should be removed and it does not connect any new link to them or does not exceed the number of the connections at some node.

4.8.4 Numerical results - expansion

N	$T[sec.]$	$C_{ost} = 41$	$C_{ost} = 51$	$C_{ost} = 60$	$C_{ost} = 70$
10	14.8	97	0	3	0
12	24.7	81	0	19	0
14	48.6	87	2	11	0
16	58.7	69	13	15	3
18	85	56	28	9	7

Table 4.7: Network expansion - results

It is possible to see the numerical results of the network expansion in the Tab. 4.7 (setting of the algorithm and input parameters are in the appendix 6.2.0.6). The first column corresponds to the number of nodes of the original network. The second column corresponds to the time that is necessary for the network expansion. The third column corresponds to the number of results which acquisition costs are 41 (optimum). The fourth column corresponds to the number of results which acquisition costs are 51. The fifth column corresponds to the number of results which acquisition costs are 60 and the last column corresponds to the number of results which acquisition costs are 70.

The algorithm should design the network expansion for two added node. The original network as well as the expanded network should meet request of two-fault-tolerance (network is possible to interrupt at two different places and the network is still operational). It is possible to use two different physical layers for the physical topology design: Metallic and fibre cables. Every node has three communication ports of both physical layers and the original network is designed only with the application of the metallic cables. The cost matrix looks as follows; every element of the cost matrix for the metallic cables is equal to 1. Every element of the cost matrix for the fibre cables is equal to 10 and elements of the cost matrix for removing of every kind of cable is equal to 100.

The parameter settings described above means that there is no possibility to connect a metallic cable to the nodes of the original network without removing of some original communication link. It means that we are able to reach the smallest acquisition costs only with the addition of the new links into the network. The minimal number of the communication links that must be added for connection of the two new nodes and keeping of the $2-fault-tolerance$ is 5 and only one communication link can be from the metallic cable (other four communication links are connected to the original network that has node with no metallic communication port available. Thus, the metallic cable can connect only newly added nodes). Therefore, the minimal acquisition costs of the expanded network are 41. The four links that are connected directly to the nodes of the original network are from the fibre cable and one communication link that interconnects the new nodes is from the metallic cable. This configuration ensures that the new network is $2 - fault - tolerant$ and its expansion is as less expensive as possible.

The results in the table were gained for a hundred of launches for every number of the nodes of the original network. Parameters of the genetic algorithm were set as follows; the number of chromosomes is 100 as well as the number of iterations. Probability of the mutation and crossover is $p_{mut} = 0.2$ and $p_c = 0.3$ respectively. Cost matrices were described above.

We can see the influence of the number of chromosomes and number of nodes of the network on the results quality. We can see that the rate of the best result ($cost = 41$) is decreasing with the number of nodes of the network (the number of chromosomes are the same) and rate of the worse solutions increases. Thus, it is possible to assume that the number of chromosomes for the network of 20 nodes should be bigger than only 100. We can see also that the time necessary for the solution of the network expansion increases with the number of nodes. It is a natural attribute of the algorithm since with the increasing of the number of nodes, the number of the node-pairs for which is necessary to verify the number of the independent communications paths increases as well. We can see that the time that is necessary for the network expansion corresponds to the time that is necessary for the design of the network with the same number of nodes (Tab. 4.3). It was possible to expect this kind of results since the core of the algorithm for the design of the fault-tolerant network and for the network expansion is the same.

4.8.4.1 Numerical results - reduction

N	$T[sec.]$	$C_{ost} = 603$	$C_{ost} = 622$	$C_{ost} = 632$	$C_{ost} = 641$	$C_{ost} = 651$
12	9.5	33	62	0	5	0
14	17.3	19	66	0	15	0
16	28	6	75	0	19	0
18	43.9	9	71	2	18	0
20	65.5	4	44	11	32	9

Table 4.8: Network reduction - results

It is possible to see the numerical results for the network reduction in the Tab. 4.8 (setting of the algorithm and input parameters are in the appendix 6.2.0.7). The first column corresponds to the number of nodes of the original network. The second column corresponds to the time that is necessary for the network reduction. The third column corresponds to the number of results which acquisition costs are 603. The fourth column corresponds to the number of results which acquisition costs are 622. The fifth column corresponds to the number of results which acquisition costs are 632. The sixth column corresponds to the number of results which acquisition costs are 641 and the last column corresponds to the number of results which acquisition costs are 651.

The algorithm should reduce the network by two nodes. The original network as well as the reduced network should meet requirements for the two-fault-tolerances. The algorithm can use two different physical layers (metallic cable and fibre cable). Every node has three communication ports of every kind of the physical layers. The original network was designed only with the application of the metallic cables. The cost matrix looks as follows; every element of the cost matrix for the metallic cables is equal to 1. Every element of the cost matrix for the fibre cables is equal to 10 and the cost matrix for removal of every kind of cable is equal to 100. All communication links that are connected to the node, which should be removed, should be removed as well.

The setting described above means that minimally 6 communication links must be removed and minimally 3 communication links must be added. Thus, the minimal acquisition costs are 603.

The results in the table were gained for a hundred of launches for every number of nodes of the original network. Parameters of the genetic algorithm were set as follows; the number of chromosomes is 100 as well as the number of iterations. Probability of the mutation and crossover is $p_{mut} = 0.2$ and $p_c = 0.3$ respectively. Cost matrices were described above.

We can see that the rate of the best result ($cost = 603$) decreases with the number of nodes of the network. The time corresponds with the time that is necessary for the design of the network with the same number of nodes.

We can see that the proposed algorithm is able to not only design the brand new network but also to design a reduction or an expansion of the existing network. The algorithm is able to reduce or expand not only the number of the communication links but also nodes of the network. Other algorithms [3],[23] are not able to do so. The proposed algorithm can work with a priori information about unwanted topologies and limitation of the number of communication ports at every node in contrary to other algorithms.

4.9 Algorithm settings

The important attributes of the genetic algorithms are their settings. Several parameters can be set. It is possible to set the probability of the mutation and the crossover. Other parameter is the number of chromosomes and stop criterion. They have an influence on the number of the chromosomes that are created during the genetic algorithm iteration. The stop rule is the number of iteration in this case. Therefore, the stop rule has the direct influence on the number of the newly created chromosomes during the whole run of the genetic algorithm. If the number of chromosomes increases and other parameters are the same, the number of the newly created chromosomes during the run of the algorithm increases as well. If the probability of the crossover and the mutation increase and the number of chromosomes is the same as well as the stop rule, then the number of the newly created chromosomes during the run of the whole algorithm increases as well.

The number of chromosomes that are created for the purpose of the genetic algorithm has an influence on the coverage of the search space. If the coverage is more precise, it is likely to gain better results than for less precise coverage. Therefore, it is important to find an influence of the number of chromosomes in the initial population on the result quality for the big and medium sized networks.

The number of mutations, crossover operations or the number of chromosomes also has the direct influence on the time that is necessary for the run of the genetic algorithm. The more chromosomes of genetic operations the longer time for the run of the genetic algorithm needs. This is not a big issue for the small network that have 15 nodes or less but for bigger network the design can be time demanding. Therefore, it is necessary to find the way to speed up the genetic algorithm.

If we look at the time complexity of the genetic algorithm, we find that the evaluation of the chromosome needs the biggest part of the time necessary for the algorithm run. The evaluation itself is not so time demanding as the verification of the number of the redundant communication paths. The Ford-Fulkerson algorithm is used for the verification of the number of the independent paths.

A parallelization of the genetic algorithm allows to speed up the whole algorithm and find suitable physical topology faster than without parallelization. Theoretically, if 4 computers are used for parallelization, the time necessary for the run of the algorithm will be quarter of the original time. If eight computers are used, the time necessary for the run of the algorithm should be eighth of the original time. Unfortunately, this is not possible because the Amdahl's law is valid and the parallelization needs time for the communication and management of the parallelization.

There are three basic possibilities of the genetic algorithm parallelization [32].

The first solution and the simplest: the task is divided into the same number of parts as computers and the task runs independently and only the results are compared. The implementation of this solution is the simplest one since it is not necessary to do almost any change in the algorithm but there is also a disadvantage. The computers do not share the results among them and therefore there is no possibility so that one computer and its task have an influence on the tasks in other computers. Then we lost one advantage of the parallelization, because every computer works only with the chromosomes that are in its task and do not have any information about chromosomes in other computers.

Other possibility is to divide the whole genetic algorithm among the computers but the computers share part of the results. The genetic algorithm sends after every iteration the best results to other computers and their tasks. Thanks to that, every task can work with the best results from the previous iteration and can use it for a creation of better descendants. A disadvantage of this solution is that the tasks in the different computers must be synchronized and all tasks wait for the slowest one. The solution time at every computer can vary even if the whole task is scheduled properly and tasks at all computers need the same time for an iteration at the start of the algorithm run. Other disadvantage is the same as for previous way of the parallelization. If we solve a difficult problem or want to get results fast, we divide the population among many tasks at separate computers. If we split the chromosome population among too many tasks, in one task there will be as few chromosomes that the crossover operator can lost abilities.

The last possible solution is to distribute only the most time demanding part of the algorithm and do the rest operation only at one central station that also manages the distribution to other tasks. The most time demanding part is the chromosome evaluation. The division of the chromosome evaluation among computers allows to speed up the genetic algorithm and genetic operators can work with the whole population of the chromosomes. This is an advantage of this solution since the whole chromosome population has an influence on the offspring created by the genetic operators. The disadvantage is that it is necessary to solve synchronization of computes at which the genetic algorithm is launched. Other important task is a division of the load among the computers. Even if computers are the same and they have the same performance, it is necessary schedule the number of chromosomes that are sent from the "Master" computer to the "Slave" computers. The computers performance is dependent not only on the hardware but also on the software. It is not very probable, that at every computer, which solves the network design, is the same installed software. Therefore, the scheduling of the chromosome distribution must be solved.

The small task is launched at every computer at the start of the parallelized design of the network. The task is to design the network with the different fault-tolerance in the different parts of the network. Every computer that has the software for the distributed design sends the time that is necessary for the task solution. If there is a computer that should be a "Master" computer (computer that does all genetic operations and manages the chromosome sharing), it receives this time from every computer in the network. The slave stations are locked only for the "Master's" task and "Master" computer calculates size of the chromosome portion for every computer. The time, which needs every computer for the solution of the test task, is transformed into the number of algorithm iterations per second. For the master computer, the number of the iterations is divided by 2. Then, the number of chromosomes is split up among the computers according to number of the iterations per second of every computer. This chromosome proportion is stable during the whole design of the network and does not vary. It means that if some computer starts solving some other task that takes the performance from the task of the network design, the whole network design slows down since the "Master" computer must wait for the slower computer.

The procedure of the parallelization of the genetic algorithm is depicted symbolically in the Fig. 4.41. We can see that the only chromosome evaluation is distributed among computers. The genetic operations are done only at the "Master" computer as well as the verification whether the stop rule is satisfied. Even if only the evaluation of the chromosomes is parallelized, it is possible to save the time necessary for the solution of the network design problem. The setting of the algorithm is as follows: The number of chromosomes is 100 as well as the number of the algorithm iterations. The probability of the crossover and mutation is $p_c = 0.2$ and $p_m = 0.05$ The computers at which ran the algorithm were the same: Intel Q87300, 2,49GHz, 2,96 GB; WinXP-SP2. The algorithm was written in the C# .NET2.0.

We can see the part of results that were gained by the parallelization of the genetic algorithm in the Tab. 4.9 and Fig. 4.42. It is possible to say that the performance of the parallel algorithm grows with the number of computers working at this task. We can also see that the increase of the performance, if the number of computers grows from 1 computer to 2 computers, is bigger than two times for the network with 30,

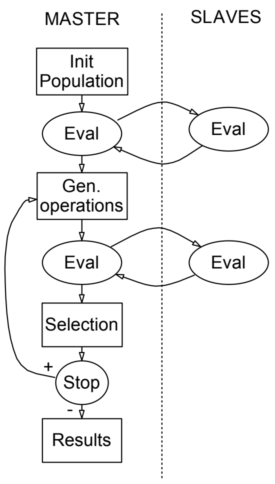

Figure 4.41: Parallel algorithm

$Computers$	1	2	3	4
$Nodes\ N$	$Time[sec.]$			
20	76.7	39.8	33.5	34.4
30	310	120	80.4	67.2
40	873.8	317.4	190.7	146.4
50	1921	703	425.3	300

Table 4.9: Parallelization - time consumption

40 and 50 nodes. This should not be possible since all computers were the same and therefore the performance increase should be close to two but it should not be bigger than two. The results are not accidental since there have been 20 launches for every configuration and results for the same configuration were almost the same. The 2 times increase of the performance is caused by the scheduler of the operation system. If the task needs too much time, then the scheduler decreases the priority given to the task and then the time for the algorithm solution increases. We can see that for the network with 20 and 30 nodes, the increase of the performance corresponds to the increase of the number of computers (time necessary for the solution with the usage of 2 computers is approximately 2 times bigger than time necessary for the solution if 4 computers are used).

We can see detailed graph for the network with 20 and 30 nodes in the Fig. 4.43. It is possible to see there other important attribute of the parallelization. It is the general behaviour of the parallelization not only the behaviour of the parallelization of the network topology design. The increase of the parallelized algorithm performance is limited. The addition of the computers can actually decrease the performance of the parallelized algorithm. The performance of the algorithm with 6 computers is worse

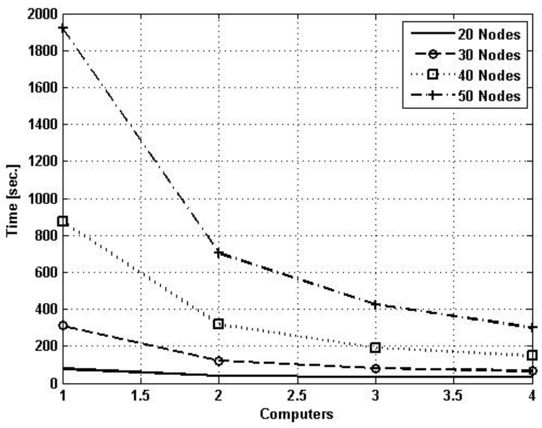

Figure 4.42: Algorithm performance

than performance for the algorithm with only two computers for the network with 20 nodes. This performance decrease is caused by the increase of operations that must be done for the communications among the computers. Master and slave computer exchanges not only the chromosomes but also acknowledgment about chromosome receiving during every iterations. The communication among the computers and its management increases with the number of the computers so fast that it is able to slow down the whole design of the network topology if the task for every computer is too small. Therefore, it is always necessary to consider whether the increase of the number of computers can increase the algorithm performance. This is a general behaviour of the parallel algorithm, increasing of the number of computers over certain limit does not bring any performance increase since the management of these computers needs more time than the additional performance can bring.

Characteristics described by the tables and graphs are valid for this specific case, for other cases the limitation of the number of computers is different and it is necessary to find it for every specific task and parallel algorithm.

It is possible to see dependency of the time, which is necessary for the network design for the network with 40 nodes, on the number of computers and number of chromosomes in the Tab. 4.10.

The first column is the number of chromosomes; the second is the time, which is necessary for the solution of the network topology design. The third column is the number of computers; the fourth column is the size of chromosomes that are exchanged during an iteration of the algorithm. The results were gained with the same settings of the algorithm and the same computers as previous results. The time is the average time necessary for the solution of the task. The average time was counted from a hundred launches of the algorithm. We can see that the parallel processing can bring important time saving. It is possible to get results for 800 chromosomes in the

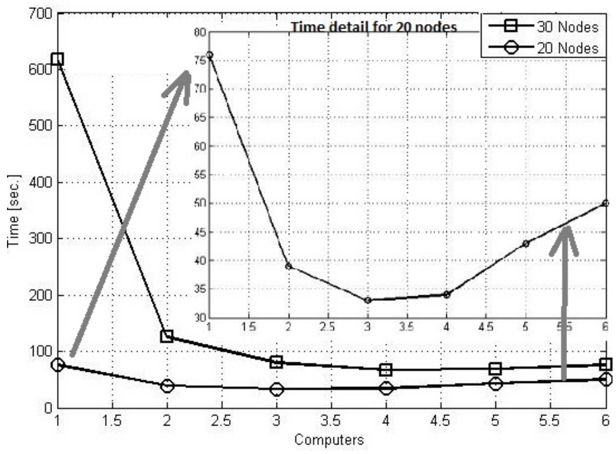

Figure 4.43: Algorithm performance - detailed view

$NofChromos$	$Time[sec.]$	PCs	kB
200	290	4	19
300	390	4	28
400	502	4	38
500	519	5	38
600	520	6	38
700	544	6	44
800	556	7	43
900	585	8	43

Table 4.10: Parallelization - time consumption

time longer only for a minute than result for 400 chromosomes. On the other hand, it is necessary to use two times more computers. It is possible to expect that will be possible to solve a design of the network with hundreds of nodes thanks to the parallel processing of the algorithm and bigger performance of computers in future.

The number of chromosomes has an influence on the quality of the results and on the speed of the algorithm as well. Unfortunately, for the finding of the good result is necessary to launch algorithm for longer time with bigger number of chromosomes. In general, it is possible to say the bigger time of the solution the better solution is possible to find. It is necessary to find the suitable number of chromosomes for the size of every task (search space). The number of chromosomes should allow to find a good result in reasonable time. This number must be found since it is not known any general procedure to find the setting of the genetic algorithm. Unfortunately, it is not possible to do it analytically. Thus, the series of tests for the network with 10, 20, 30, 40 and 50 nodes were done. If we look at the Tab. 4.4, we can see that the setting of the algorithm was suitable since the most often found result corresponds to the minimal acquisition costs. Thus, the $N_{ch} = 100$ with the applied probabilities of the crossover and mutation is the good setting for the network with $N = 10$ and it is not necessary to search for better settings. On the other hand, for the network with 20 nodes, the number of chromosomes allows to find solutions that are close to the optimal one but any solution does not correspond to the minimal acquisition costs that are 25. The number of chromosomes $N_{ch} = 100$ allows to find the solutions that are mostly better than the algorithms that are not able to find the network with the different levels of the fault-tolerance and they must design the network with the same level of the fault-tolerance in the whole network. The minimal acquisition costs for these algorithms are $c_{ost} = 30$, which is more than the average value reached by the algorithm that is able to design the network with the different levels of the fault-tolerance in the different parts of the network. We understand as appropriate number of chromosomes such number that allows to find better solutions than it is the best solution of the algorithm that is not able to design the network with the different levels of the fault-tolerance in the different parts of the network.

The algorithm solves the following task of the network design. The network should have different levels of the fault-tolerance; the 1-fault-tolerance is required in the bigger part of the network and the 2-fault-tolerance is required in smaller part. It means that algorithm, which is not able to design the network with the different levels of the fault-tolerance, must design the network that meets requirement of the 2-fault-tolerance in the whole network. It means that if the every communication link costs 1 unit, then the minimal acquisition costs of the network designed by the algorithm that is not able to design the network with different levels of the fault-tolerance in the different parts of the network are:

$$c_{ost} = 1,5N. \tag{4.47}$$

The number of nodes must be even; the node pair in the network, which must meet $2 - fault - tolerance$, must have 3 independent communication paths. It means that the minimal configuration of the network must be similar as the network depicted in the Fig. 4.27. The number of pair-nodes that should have 3 independent communication paths is as follows:

$$j = \frac{(N-10)}{2} + 4. \tag{4.48}$$

Knowledge about the number of the communication links in the network allows to count the minimal acquisition costs of the network designed by the algorithm. Then the minimal acquisition costs are:

$$c_{ost} = Round\left(1,25N\right). \tag{4.49}$$

It is possible to set boundaries of the acquisition costs of the ”good” designed network thanks to the knowledge of the minimal acquisition costs of the network, which meets demands for the fault-tolerance requirement, designed by our algorithm and algorithm that ensures the same level of the fault-tolerance in the whole network. The average acquisition costs should be between the values gained by the equation (4.47) and (4.49).

The results must have statistical character since the genetic algorithm may find the optimal solution at the first launch with very small number of chromosomes and solutions of other launches could be much worse. Therefore, it is necessary to do series of launches for every setting and after then it is possible to say that the algorithm is able to find the result that is statistically better than solutions of other algorithm. Importance of the finding of the appropriate number of chromosomes is possible to see in the Tab. 4.11 where the size of the search space is.

N	$Space1PHL$	$Space2PHL$
10	$3.5E + 13$	$2.95E + 21$
15	$4.05E + 31$	$1.25E + 50$
20	$1.56E + 57$	$4.49E + 90$
30	$8.87E + 130$	$3.52E + 207$
40	$6.35E + 234$	$1.4E + 372$
50	$5.7E + 368$	$2.9E + 584$

Table 4.11: Size of search space

In the first column is the number of nodes of the network. In the second column is the size of the search space for the network that can use only one physical layer for the network construction. In the last column is the size of the search space for the network that can use two different physical layers for the network construction. If we look at the size of the search space, it is obvious that the basic number of chromosomes equal to 100 cannot cover the search space of the network with 50 nodes.

In the Tab. 4.12 is possible to see the recommended number of chromosomes for the design of the network with the different number of nodes and the average acquisition cost that were reached for the network designed by the proposed algorithm for the design of the network with the different levels of the fault-tolerance. The dependencies for the different number of nodes and chromosomes are possible to see in the graphs in the appendix.

N	N_{ch}
10	100
20	200
30	500
40	1500
50	5000

Table 4.12: Recommended number of chromosomes

The data in the Tab. 4.12 were gained after numerous tests for the different sizes of the network and number of chromosomes for the same settings of the algorithm, where $p_c = 0.25$, $p_m = 0.05$ and number of iterations 100. Every node pair should have minimally two independent communication paths and $\frac{(N-10)}{2} + 4$ node pairs should have minimally 3 independent communication paths. The optimal number of chromosomes was assumed to be such number that allows to gain the average results:

$$c_{ost} = Round\,(1,25N) + 0,76(1,5 - 1,25)N. \tag{4.50}$$

The statistical results were gained minimally for a hundred of runs of the algorithm. The setting described above ensures that algorithm for the design of the network topology with the different levels of the fault-tolerance in the different parts of the network gains statistically better results than the algorithms that use a node degree as a fault-tolerance measure (for them every node pairs must have $\deg(v) = 3$ if $2 - fault - tolerance$ is needed in the part of the network).

The recommended number of chromosomes was gained for the task described above but it is possible to assume that this setting allows solution of the different tasks in which the different levels of the fault-tolerance is needed as well. The size of the search space is the same for the every task with the same number of nodes of the network independently on the number of communication paths. Therefore, the probability of the optimum finding is the same. Thus, it is possible to expect that the recommended number of chromosomes allows to reach the similarly successful results for the different tasks as well.

4.10 Design of tree topology

The mostly applied topology in the control engineering is the tree topology that is easier to design than the network with the different levels of the fault-tolerance in the different parts of the network.

There have been published papers that focus on the design of light paths in the optical networks [35],[36] or computer networks [37] or communication restoration algorithms that use backup trees for communication recovery if some communication link fails [38]. Other algorithms were focused on design tree based networks in parallel computing [39] or on the reliable communication for data-collection systems [40]. Another application of the tree topology is possible to see at multicast tree for video broadcasting [41].

It is simple to design the network with the tree topology if we want to design the network with the smallest acquisition costs. There are many algorithms that are able to find the minimum spanning tree (Prim's, Kruskal's, Dijkstra's algorithm [14]) or just spanning tree. It is easy to modify them if we want to use the different physical layers and still want to find the network with the smallest acquisition costs as possible. The solution is not so straightforward if we want to use a priori information about unsuitable topologies and constrain the depth of the designed topology. Algorithm (Prim's, Kruskal's) finds the cheapest network. If we want to use them and the tree should have constrained depth, it is necessary to use some transformation from the depth of the graph to the acquisition costs. This transformation cannot ensure a finding of a good result in every case since the result is dependent on the cost of every communication link and the transformation function should be chosen for every design.

The algorithm that finds the shortest tree is described in [42]. The algorithm uses the linear programming for finding of the Steiner tree. Algorithm is not able to use a limitation of the number of the communication ports and a priori information about the unwanted topologies. Therefore, it is not convenient for the solving of the issue of the design of the tree topology with attributes mentioned above and help of a priori information.

The design of the tree topology is used in the VLSI circuit design. The method is described in [43]. The method uses a heuristic algorithm that does not work with a priori information about unwanted topologies or with the limitation of the depth of the tree. On the other hand, the algorithm considers capacity assignment as well as the node degree limitation.

The method described in [40] designs the tree topology for the sensors network. There are precisely described abilities of the tree topologies with the different depths. The algorithm is based on the modified Prim's algorithm [14] that uses the transformation of the depth of the communication links into the cost function. This allows to find the topology that incorporates the depth into the price of the network. Nevertheless, the algorithm does not work with the limitation of the node degree and does not use a priori information about the unwanted topologies as well. Therefore, it is not suitable for the application in the control engineering (every node has a limited number of the communication ports that have limited possibility to be expanded).

The method described in [44] proposes an original encoding of the chromosome for the genetic algorithm for the design of the tree topology of the LAN network. The encoding allows to use the smallest possible size of the memory for the storing of all chromosomes. However, the algorithm does not work with the limitation of the number of communication ports or depth of the network. The algorithm does not use a priori information of the unwanted topologies and therefore it is not suitable for the design of the network in control engineering.

There were proposed two methods for the design of the tree topology in [45]. The first of them is based on the simulated annealing method and another on the Integer Linear Programming. Both methods work with the limitation of the node degree and the maximal depth of the tree topology. Described methods are possible to use for the finding of the tree topology if it is not necessary to use a priori knowledge to prevent unwanted topology design.

Hence, the method for a design of the network with the tree topology is proposed in this document. The network is designed with the help of the genetic algorithm that can easily incorporate the limitation of the depth of the network as well as the limitation of the number of the communication ports. Moreover, the genetic algorithm allows to use a priori knowledge of the unsuitable topologies that are not possible to use from some reason (e.g. the limitation of the environment in which the network will be used). It is necessary to use an appropriate representation of the network as well as genetic operators and the fitness function for the application of the genetic algorithm.

4.10.1 Chromosome representation - Tree topology

The representation of the network topology is the same as it is described in the chapter 4.8.1 **Different physical layers**. This representation allows to design the network topology with the application of the different physical layers. This representation is able to describe every kind of the network topology and does not provide any advantage for the description of the tree topology. Therefore, it is necessary to verify whether the chromosomes really represent the tree topology.

4.10.2 Genetic operators

4.10.2.1 Initial population

It is necessary to create the initial population of the chromosomes for the application of the genetic operator. The quality of the initial population has an influence on the speed of the convergence of the genetic algorithm [12]. Therefore, the initial population is split into two parts: The first that corresponds to the 75% of the initial population describes the network with only the tree topology and the second part (25%) contains chromosomes that represent the general topology.

In the beginning, the first part of the initial population is created at random and then the repairing algorithm is used for the verification whether the randomly created topology is tree. If not, the reparation procedure repairs chromosome to be the tree. It is ensured that the number of communication links at any port in the designed network topology is not bigger than the actual number of the communication ports at the node during the random design of the chromosome. The reparation algorithm is a heuristic algorithm that reconfigures the chromosome to describe a tree that meets demands for the maximal depth of the tree. The reparation procedure works as follows:

1. Find unconnected nodes and connect them together via the physical layer that is the cheapest.

2. Interrupt all cycles in the network (remove the communication link that closes the cycle).

3. Create the list of the nodes that are start nodes of the parts that are not connected to the parent node of the tree.

4. Connect the nodes from the list to the nodes that have available free communication port and they are as close as possible to the parent node.

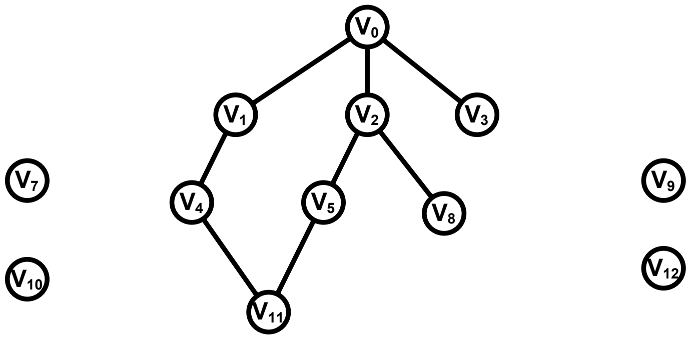

Figure 4.44: Tree topology - initialization

It is possible to see the result of the random chromosome initialization in the Fig. 4.44. We can see that network contains cycle and unconnected nodes. Therefore, it is necessary to use repairing algorithm described above. All nodes have 4 communication ports in total (for purpose of the explanation, the communication ports are the same type as the physical layer is). The result of the repairing algorithm is possible to see in the figure Fig. 4.45.

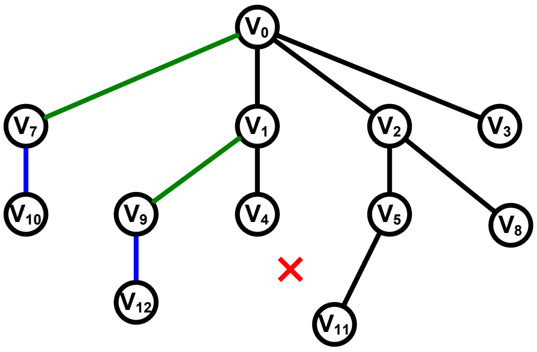

Figure 4.45: Tree topology - After repairing

It is possible to see in the Fig. 4.45 not only the result of the application of the repairing algorithm to the network depicted in the Fig. 4.44 but also the advance of the repairing algorithm. The first step of the repairing algorithm is depicted with the blue colour, the second step with the red colour and the last step with the green colour.

During the first step, all unconnected nodes are connected together. Then every node is connected to the other node minimally via one communication link. In the second step, all cycles in the network are disconnected (the link between nodes V_4 and

V_{11} is removed). The bread-first search algorithm finds cycles in the network. Then in the last step, all component that are not connected to the network are connected to the network as close to the root node as possible. The node V_7 is connected directly to the root node since the root node has unused communication port available. The node V_9 is connected to the node V_1 since there is any other node closer to the main node. All chromosomes, which are passing through the repairing procedure, are repaired in this way.

The reparation procedure ensures that the repaired chromosomes describe the tree topology and do not violate the limitation of the communications port at some nodes.

The rest of the initial population is created at random; it means that the topology can include cycles or unconnected parts of the network. This part of the initial population is created to provide the chromosome diversity that can help to reach better results during the run of the genetic algorithm thanks to the genetic operators.

4.10.2.2 Mutation operator

The common one bit mutation [12] is used. The chromosome that is going to be mutated is chosen at random as well as the place of mutation. Then the gene is mutated in the same manner as it is described in the section 4.8.1 **Different physical layers**. After the application of the mutation operator, it is necessary to verify whether the chromosome describes the tree topology and repairs it if not. The reparation of the chromosome is done via the repairing algorithm described above.

4.10.2.3 Crossover operator

The one point crossover operator [12] is used. The pair of chromosomes that is crossed is chosen at random as well as the place of crossing. The crossover operator can create offspring that do not describe the tree topology. Thus, it is necessary to use the repairing algorithm described above. The reparation algorithm ensures that offspring created by the crossover operation are tree topologies that do not violate the limitation of the number of the communication ports.

4.10.2.4 Fitness function

The fitness function is crucial for the evaluation of the chromosome quality. The fitness function must evaluate the chromosomes and ensures that any chromosome that somehow violates some limitation does not have better evaluation than permissible one. The fitness function must be able to evaluate acquisition costs of the network described by the chromosome and also penalize the chromosome. The penalization is necessary if the depth of the network is bigger than permitted or if there are bigger demands for the communication ports than there is the available number of the communication ports of the devices or if the chromosome describes the network that is unsuitable. The fitness function is the sum of the acquisition costs of every communication link applied in the network and the penalty function is as follows:

$$\begin{aligned} Pen &= k\left(N^2+1\right)+k_{dis}N^2+\left(u+k_{cyc}\right)N^4+ \\ &+N^3\left(Depth-Depth_{\max}\right)|_{Depth>Depth_{\max};\forall v}\,. \end{aligned} \tag{4.51}$$

Where k_{dis} is the number of nodes that are disconnected from the main node. k_{cyc} is the number of cycles in the network. The last element $(Depth - Depth_{max})$ is used for every node that is in bigger depth than it is the maximal permitted depth of the tree. u is the element that is set to 1 if the network is the same as some of unwanted topologies, in other cases the element is 0.

We can see that there are elements in the penalty function that seem to be useless as k_{dis}, k_{cyc} and k. These elements are not applied for the chromosomes created by the genetic operators or for chromosomes belonging to the 75% of the initial population, all these chromosomes are repaired by the repairing procedure. On the other hand, there is 25% of the chromosomes in the initial population that are created at random and these chromosomes can contain description of the cycles, unconnected nodes or may exceed the limitation of the maximal communication ports at some nodes. Therefore, all characterization of above mentioned elements are in the penalty function.

4.11 Reduction/expansion of the tree network topology

It is often necessary to change the topology of the existing network. Mostly it is because of the changes in the controlled technology. It is necessary to change topology if the node is added or removed from the network. The genetic algorithm is used for the design of the tree topology as a key stone of the algorithm for the reduction/expansion of the network with the tree topology.

The algorithm for the network reduction/expansion must be able to redesign the network in order to meet the same limitation as the brand new network. Therefore, the algorithm must be able to design the network with the tree topology with the limited depth. The number of connections of the node in the network must not exceed the number of communication ports. Moreover, the algorithm should be able to use a priori information about the unwanted topologies.

It is important to use knowledge of the original topology during the network reduction/expansion and not to design the network from scratch since the knowledge of the original network can help to reach better results. Therefore, the topology of the actual network is used during the creation of the initial population of the genetic algorithm.

4.11.1 Tree topology expansion

The expansion of the tree topology must be able to deal with the increase of the number of nodes and the expansion should be as less expensive as possible. The genetic algorithm is used for the network expansion. The algorithm recognizes that the network should be expanded according to the size of the matrices of acquisition costs and array that describes the number and the kind of the communication ports. The knowledge of the original topology is used for the creation of the part of the initial population. 25% of the initial population is created on the basis of the original topology.

4.11.1.1 Creation of initial population

The 25% of the initial population is created according to the algorithm:

1. Create chromosome that is able to code the whole network including the added nodes and describes the original network.

2. For the chromosome part that describes the original network: Randomly decide if the gene is going to be changed. If yes, randomly choose if there will be a connection of the nodes and a material of the connection or if the existing connection is going to be removed. The connections can be added only until the limitation of the communication ports is reached.

3. For the chromosome part, that describes the new part of the network: Randomly decide for every gene if there will be a connection and type of the connection that is described by the gene. It is possible to add connections only until the limitation of the number of communication ports is reached.

4. Launch the repairing algorithm described above in order to repair the chromosome to describe the tree topology.

5. Repeat steps 1..3 until the 25% of the initial population is created.

Moreover, the greedy algorithm creates five chromosomes in order to find the cheapest configuration of the node connections. This configuration must meet only the condition for the maximal number of the communication links of every ports. The number of connections must not be bigger than the number of available communication ports. The topology can have arbitrary depth.

The rest of the initial population is created at random in order to create bigger variety of the population.

4.11.1.2 Genetic operators

Genetic operators work in the same manner as it is described in the section 4.10.2 **Genetic operators for the design of the tree topology**. The repairing algorithm is launched for every result of the genetic operators.

4.11.1.3 Fitness function

The fitness function is an important part of the genetic algorithm and has an influence on the result quality. The fitness function must be able to evaluate a quality of the chromosome as well as a violation of the limitation of the network topology.

The fitness function contains two parts. The first part evaluates the differences between the newly designed topology and the original topology. The second part penalizes chromosome if the chromosome does not meet some requirements. The first part works similarly as the fitness function used for the expansion of the network with the different levels of the fault-tolerance in the different parts of the network. The penalty function is the same as it is described for the design of the tree topology, then:

$$c_{ost} = \sum c_{e^+} + \sum c_{e^-} + Pen. \tag{4.52}$$

Where c_{e^+} is the cost of all edges that were added to the original topology and c_{e^-} is the cost of all edges that were removed from the original topology. It is possible to find the all removed or added edges thanks to the comparison of the created chromosome with the mask created according to the original "adjacency" matrix. The creation of the mask is described in the chapter 4.8.3.1 **Network expansion**.

4.11.2 Tree topology reduction

It is often necessary to reduce the network if the controlled technology is changed. The reduction of the network should be as less expensive as possible and the network should meet the requests as the depth of the tree and a maximal number of connections of nodes. Moreover, the topology should not be the same as some of the unwanted topologies since there can be limitation in the environment in which the network is used. Therefore, the genetic algorithm is used since it allows to meet all demands at the network topology.

4.11.2.1 Initial population

Knowledge of the original network is used for creation of the initial population. 25% of the initial population is created with the help of this knowledge according to the following algorithm.

1. Create the chromosome that describes the original topology.

2. For every gene in chromosome: Verify, whether gene corresponds to the connection to the node that should be removed. If yes set the gene to 0 and continue with other genes. If no, continue to step No. 3.

3. Randomly decide if the gene will be changed. If yes, randomly choose change of the gene (the connection is possible to add only up to the number of communication ports of certain type).

4. Launch the repairing algorithm for the tree network reduction.

The repairing algorithm ensures that every chromosome, which passes through it, describes the tree topology and every node in the network does not exceed the maximal number of the connection. The repairing algorithm does not ensure the maximal depth of the network.

The crossover operator is the same as for the network expansion. The mutation operator is one bit mutation. The place of mutation is chosen at random; the place of mutation can be every place in the chromosome excluding the place that corresponds to the connection to the node that should be removed. Every resulting chromosome must pass through the repairing algorithm that ensures that the chromosome describes the tree topology. The fitness function is as follows:

$$c_{ost} = \sum c_{e^+} + \sum c_{e^-} + Pen \tag{4.53}$$

and the penalty function is:

$$\begin{aligned} Pen \; = \; & k\left(N^2+1\right) + k_{dis}N^2 + \left(u + k_{cyc}\right)N^4 + \\ & + N^3\left(Depth - Depth_{\max}\right)|_{Depth>Depth_{\max};\forall v} \, . \end{aligned} \tag{4.54}$$

The meaning of elements was described at previews equations. The element e_r must be in the fitness function since the part of the initial population is created at random and therefore the node, which should be excluded, can be connected via some communication link.

It is possible to do the reduction and the expansion of the network at the same time if we combine the above described algorithm similarly as the algorithm for the reduction and the expansion of the network with the different levels of the fault-tolerance in the different parts of the network (see section 4.8.3.3).

4.11.3 Numerical result - tree topology

The results of the design of the tree network topology are described in this section. The results of the network topology expansion and reduction are described as well.

N	Min	$T[sec.]$	$0link$	$1link$	$2link$	$3link$	$4link$	$5link$	$6link$
10	36	3.16	9	37	54	0	0	0	0
12	47	4.58	4	57	39	0	0	0	0
14	32	5.78	0	2	32	66	0	0	0
16	44	7.25	0	2	36	58	4	0	0
18	44	8.9	0	0	5	5	44	43	3
20	55	10.6	0	0	3	19	40	32	6

Table 4.13: Design tree topology

4.11.3.1 Design of tree topology

There are results gained by the proposed algorithm in the Tab. 4.13. The setting of the algorithm is as follows: $p_c = 0.3$, $p_m = 0.02$. The number of chromosomes was 250 and the number of iterations was 100. It was possible to use two different physical layers in the network and every communication node has four communication ports of every kind available.

The first column is the number of nodes in the network. The second column is the minimum possible acquisition costs that are possible to reach. The third column is the time that is necessary for the design of the network with the tree topology that contains the number of nodes from the first column. The fourth column means how often the algorithm found the topology in which 0 links are different from the optimal placing. The fifth column means how often the algorithm found the topology in which is exactly one link different from the optimal placing. The sixth, seventh, eight, ninth and tenth column describes how often the algorithm found the topology that differs exactly in two, three, four, five and six link position from the optimal topology respectively.

The physical topology should contain two different physical layers and price matrices exactly describe how the network should look like. If there should be the link from the metallic cable, the element of the cost matrix for the metallic cable is 1. If there should be the communication link from the fibre cable, the corresponding element in the cost matrix of the fibre cable is set to 10. All other elements are set to 1000. It means that there is always just the only one optimal solution and every other has worse evaluation than the optimal one.

We can see from the Tab. 4.13 that the algorithm is not able to find the optimal configuration for the network with the number of nodes higher than 12 and the setting described above. It means that there should be used a bigger number of chromosomes or different setting of the algorithm.

It is possible to see the optimal topology of the networks with the tree topologies in the section 6.2.0.8.

The optimal topology from the tree network topology design is used as the original topology that should be changed in the reduction/expansion procedure.

4.11.3.2 Numerical results of tree topology reduction/expansion

The optimal configurations of nodes and communication links for the task described above were used as an original topology that should be reduced or expanded. Settings of the algorithm were the same as for the design of the tree topology. The cost matrix for the physical layer created from the fibre cable had all elements equal to 100. The cost of the removal of the communication link was 1000 for the both physical layers. The cost matrix for the metallic cable did not have the same elements everywhere. All elements were set to 10 excluding the elements corresponding to the node $No.2$. It means that the metallic cable was preferred material for the communication links and the node $No.2$ was preferred for the creation of the new connection. Every node had four communication ports of the both physical layers that were available. Maximal permitted depth of the tree was four.

N	$T[sec.]$
10	2.38
12	3.24
14	3.78
16	4.46
18	5.06

Table 4.14: Tree topology expansion

The results of the network expansion are in the Tab. 4.14. In the first column is the number of nodes of the original network; in the second column is the time necessary for the network expansion. The costs of the expansions are not in the table since the algorithm found the optimal configuration of the added link thanks to the greedy procedure during the initialization of the population of the genetic algorithm. It means that for every number of nodes were the costs necessary for the network expansion equal to 2 excluding for the network with 16 nodes of the original network. For the network with the 16 nodes of the original network, there were the acquisition costs equal to 11 since it was possible to connect the only one new link to the node $No.2$. The node $No.2$ already had three connections to other nodes via the metallic cable and it was not possible to connect two new connections. Therefore, one new node was connected to other node via the metallic cable. The results were gained for 100 runs of the algorithm for every number of nodes of the original network. If the greedy procedure for the chromosome initialization is not present, gained results are not so good since for bigger number of nodes, the number of chromosomes is not sufficient and it would be necessary to increase the number of chromosomes.

The results of the tree topology reduction are possible to see in the Tab. 4.15. In the first column is the number of nodes of the original network; in the second column is the time necessary for the network reduction. The costs that are necessary to pay for the network reduction are not presented since the algorithm found the optimum

N	$T[sec.]$
12	2.15
14	2.78
16	3.62
18	4.11
20	4.88

Table 4.15: Tree topology reduction

in all launches of the algorithm. Every node has four communication ports of both physical layers available. The maximal permitted depth was four.

The node $No.2$ should be removed from the network; all other nodes should remain in the network. The cost matrix for the fibre cable has all elements equal to 100. The cost matrix for the metallic cable has all elements equal to 10. If the greedy algorithm is not used for the initialization of the part of the initial population, the algorithm does not find the optimal network configuration. Therefore, it would be necessary to increase the number of chromosomes. Results in the Tab. 4.15 were gained for a hundred launches for every number of nodes in the network.

Chapter 5

Conclusion

The topology of the network has the crucial importance for the attributes and the behaviour of the network. If the network topology does not allow a communication between nodes there is no way how to transfer data between nodes without change of the network topology. If the topology does not contain the independent communication paths, the network can be split into two parts that are not able to communicate between each other. The impossible communication among nodes of the network is definitely the situation that should be avoided since not only the communication is interrupted but also function of the applied control system is influenced by the interrupted communication. If the network is applied in the telecommunication, the interrupted communication can disrupt calls or data transfer. If the network is used in control engineering, the interrupted communication can cause the malfunction of the controlled algorithm and consequently the malfunction of the controlled technology. The malfunction of the controlled technology can cause not only the financial losses but also casualties. Therefore, the topology should be designed in order to prevent this situation. Moreover, the topology of the network should allow transferring data in the demanded time. The easiest solution of this issue is to create a fully-connected network in which every node is connected to all other nodes. The fully-connected network has the biggest fault-tolerance that is possible to reach and it is able to transfer the biggest load of the data. Every data-flow must pass only one communication link in order to reach the destination node and therefore the communication is as fast as possible. Unfortunately in many cases, it is not possible to design such a network since the nodes do not have enough communication ports and even if it is possible to extend the number of the communication ports, it is mostly not possible to create such a network because of the limited budget.

The limited budget forces to design the network as less expensive as possible. It means that the network topology should be as simple as possible. The construction of as simple network as possible is in contrary to the maximal fault-tolerance. Therefore, it is necessary to balance these requirements and design the network with the demanded fault-tolerance with as small acquisition costs as possible. The network is mostly possible to divide into the parts that need different levels of the fault-tolerance (the part with the critical part of the control algorithm needs bigger fault-tolerance than the part that provides the interface to public guests that are not permitted to do any change on the controlled technology). In this case, it is possible to design the topology of the network with the level of the fault-tolerance according to the highest requested fault-tolerance or it is possible to design the network that exactly meets the requested levels of the fault-tolerance. Both possibilities are described in this thesis and abilities of the designed network as well. It is possible to see from the result that the network, which has the same level of the fault-tolerance, is possible to design much faster than the network with the different levels of the fault-tolerance. On the other hand, the network with the same level of the fault-tolerance is always more expensive than the network with the different levels of the fault-tolerance (provided that is possible to split the network into the parts in which the different levels of the fault-tolerance is needed). Thus, the network with the same level of the fault-tolerance is worse than the network with the different levels of the fault-tolerance from the user point of view even if the design is much faster than the design of the network with the different levels of the fault-tolerance.

The algorithm for the design of the network topology should have also other abilities than only to design the fault-tolerant network. The algorithm should be able to design the network with an application of the nodes with the different numbers of communication ports of the different types of the physical layer since it is not possible to connect more communication links than there are available communication ports at the node. The algorithm for the design of the network topology should be able to use a priori information of the unsuitable network topologies. The application engineers often have information about the environment in which the network is going to be used. This environment can often limit possible structure of the network topology and application engineers have this information. Therefore, it is useful to use this knowledge and incorporate it into the design of the network topology. Moreover, the network must be able to accommodate the demanded traffic load and transfer this load in the requested time. The genetic algorithm that is able to design the network with the different levels of the fault-tolerance with an application of the nodes with the different number of communication ports and with a usage of prior information was proposed in this thesis. Moreover, different physical layers can be used in the designed network. The designed network allows transferring the expected load in the demanded time and verification of this ability is inseparable part of the whole algorithm.

The time that is needed for the design of the network with the bigger number of nodes (20+) is quite high. Therefore, the parallelization of the algorithm was proposed. The parallelization of the algorithm is able to decrease the time that is necessary for the network design. The speeding up of the algorithm is limited by the management of the instances of the algorithm in the network of computers. The increase of the performance is smaller than the increase of the management of another computer in the network at certain number of computers that solve the task and an addition of other computers actually decrease the performance of the whole network of computers. However, the parallelization of the algorithm was used for the finding of the recommended settings of the genetic algorithm for the network topology design. The proposed algorithm with the recommended setting is able to find statistically better results for the network that allows to use different levels of the fault-tolerance than it is the best possible result of the algorithm that is able to design the network with the same level of the fault-tolerance (The existing algorithms are able to find the network topology only with the same level of the fault-tolerance, some of them are able to work with the nodes with the limited number of communication ports and none are working with a priori information. Some algorithms are not able to find the fault-tolerant network even it seems to - see chapter 2 **Reliability**).

It is necessary to change the network topology very often if the controlled technology is changed. Existing algorithms are able to change the routing in the original network or add/remove communication links in the network. Nevertheless, these minor changes are possible to use only if the number of the nodes in the network does not change. If the number of nodes in the network is changed, the existing algorithm must design the brand new network. It means that the whole infrastructure of the network must be changed since these algorithms are not able to work with knowledge of the existing network. Therefore, the change of the network topology is more expensive than it could be. Thus, the genetic algorithm that does only necessary changes into the existing network topology is proposed in the thesis. The algorithm is able to redesign the existing network in order to accommodate new nodes. The number of changes in the network is as small as possible and network meets all demands for the different levels of the fault-tolerance. The network can contain the nodes with the different numbers of communication ports and use different physical layers. Information about unsuitable topologies can be used if it is available. Thus, the change of the network topology is as less expensive as possible and the newly redesigned network topology uses as a big part of the original network as possible.

The tree topology is often used in the industrial control systems. The big advantage of this topology is that the network is as less expensive as possible. The big disadvantage of this kind of the topology is that only one interrupted communication link makes the communication among part of nodes impossible. Nevertheless, the tree topology is used in the industry and therefore the algorithm that is able to design the tree topology was proposed in the thesis. The number and kind of the communication ports at every node is important for the design of the tree topology. Another important parameter is a depth of the tree topology since it has a direct influence on the structure of the network. If the depth of the tree is not limited, the designed topology can have structure of the bus. It is important to use knowledge about unsuitable topologies that it is possible to gain with the help of the application engineers. This knowledge can prevent an application of the network that seems to be well designed but after its application in the real environment there are many malfunctions due to interruptions caused by the external events (somebody interrupts link, or enormous heat destroyed the cable etc.). The proposed algorithm is able to use knowledge of the unsuitable topologies, kind and number of communication ports and maximal permitted depth for the design of the network with the tree topology. The network meets all demands for the maximal depth, application of different physical layers.

The network with the tree topology can be changed due to the changes of the controlled technology. Therefore, the algorithm that is able to redesign the existing network is proposed in this thesis. The algorithm is able to redesign the network even if the number of nodes in the network is changed. The redesigned network still meets all demands that were requested at the original network. The newly designed network uses as big parts of the original network as possible and thanks to that the expansion or reduction of the network is as inexpensive as possible.

The algorithms proposed in this thesis were originally created in purpose to design the network that meets all possible demands that can be requested by the application engineers of the industrial communication networks. But it is possible to use these algorithms for the design of the transportation network, pipelines or sewerage. The algorithms offer general method to design a structure that can be described as a network with the limited connections and the different numbers of the independent paths or depth for the tree topology. The algorithms are able to work with the limited capacity of the communication lines, pipes or ways. It is necessary to change the algorithm of the verification of capacity limitation of the designed network or pipelines, etc. The module that verifies the maximal time-delay of the traffic flow in the communication network is possible to change with the different module that corresponds to application of the network. This is possible thanks to the modular structure of the network. Thus, the proposed algorithms are possible to understand as general algorithms for the design of the network.

5.1 Contribution

Several algorithms were presented in this thesis. The objectives that have been set at the start of the thesis were fulfilled. The basic algorithm (chapter 4.2) allows to design less expensive network than existing algorithms if the different levels of the fault-tolerance are requested (it is necessary to know the importance of the different parts of the network in this case). This algorithm is used as a basic algorithm for the described modification. These modifications allow to design:

- *Network that uses nodes with the different numbers of communication ports of the different physical layers*. The application of the structure of the proposed algorithm and accumulator of the unsuitable solutions allows avoiding a design of the topologies that are unsuitable from some reasons (mostly because of the environment limitation). The application of the Ford-Fukerson algorithm allows to design the network with the different levels of the fault-tolerance in the different parts of the network and thanks to that meet requests for the network fault-tolerance while the costs are as small as possible. The behaviour of the network is verified during the logical topology design that allows to find the end-to-end delay in the worst-case scenario. Moreover, there is proposed the method for the estimation of the real behaviour. The method for the calculation that is used for the end-to-end delay calculation allows verifying whether every data-flow reach its destination until the maximal permitted time elapses. The method is based on the Kleinrock's interpolation that is commonly used in the literature. Nevertheless, the structure of the algorithm allows a simple replacement of the module for the end-to-end delay calculation and an acquisition of more precise results or results that are more accurate for certain type of the communication technology.

- *Expansion or reduction of the network.* The algorithm allows to design not only the addition or the reduction of the communication links as is described in the literature but also to increase or decrease the number of nodes. It means that this algorithm is able to redesign existing network if some node must be added or removed. Algorithm is able to ensure that the redesigned network meets all requests for the fault-tolerance, application of nodes with the different numbers of communication ports of the different physical layers and the usage of a priory information about unsuitable topologies since the algorithm is based on the modification of the basic algorithm mentioned above.

- *The network with the tree topology with the limited depth.* The network uses nodes with the different numbers of the communication ports of the different types of the physical layers. The algorithm is able to use a priori knowledge about unsuitable topologies that allows to avoid such topologies that are unsuitable from some reasons (e.g. environment limitation).

- *Expansion or reduction of the tree topology.* The algorithm allows to add or remove the nodes to the network. The redesigned network meets all demands for the maximal depth of the topology and the application of a priori information about the unsuitable topologies. The network can use the nodes with the different numbers of the communication ports of the different types of the physical layers.

5.1.1 Goals

The goals set at the beginning of this thesis were fully fulfilled as detailed below.

1. *To design an algorithm for the design of the network with the different fault-tolerance in different parts of the network while the network allows in time data*

delivery.

The algorithm for the network topology design is described in 4.2, 4.4 and 4.5. Results and comparisons with the existing algorithms are in the section 4.6.

2. *To modify the basic algorithm for the limited number of communication ports of nodes (different physical layers can be applied) and an application of a priori information about unsuitable topologies*

   The modification of the basic algorithm is depicted in the sections 4.8, 4.8.1 and 4.8.2.

3. *To design an algorithm for the mesh network expansion and reduction.*

   Methodology of the network expansion and reduction is described in the section 4.8.3. Both expansion and reduction are described in this section with examples of necessary modifications of the basic algorithm.

4. *To design an algorithm for the network topology design. The tree network has limited depth and use nodes with the limited number of communication ports of the different type of physical layers.*

   The algorithm for the design of the network with the tree topology is depicted in the section 4.10. The repairing algorithm as well as operators and network representation are described in this section as well.

5. *To design an algorithm for the tree topology expansion or reduction.*

   The algorithm for network expansion/reduction is described in the section 4.11. as well as modifications that are necessary for this task. In the same section, we also find numerical results of the network expansion/reduction and results of the algorithm for the tree topology design.

5.2 Future research and development

Future research is possible to split into two parts: the first is connected to the design of the physical topology and the second to the verification of the behaviour of the network

- Implementation of the limitation of the maximal number of necessary hops into the design of the network topology. Moreover, the algorithm must be able to find the optimal position of switches if the nodes do not have the switch ability. The algorithm should also prevent from designing the topology that contains long communication links since the probability of the link interruption increases with the increase of the length of the communication link.

- Improvement of the algorithm for verification of the network behaviour. More precisely, development of the scheduling algorithm for the Profinet IRT if redundancy of the communication paths is used.
- The algorithm implementation into the ”Profinet designer” that is under development at the Department of the Control Engineering at FEE, CTU in Prague.

The algorithm has enough general structure that allows to modify the algorithm not only for other communication technology but also for other parts of the industry. If the module for the verification of the network behaviour is exchanged, the algorithm will be able to design the structure of pipes or power grid.

Chapter 6

Appendix

6.1 Ability of crossover operator [1]

As it is written in the chapter 2 **Reliability**, the algorithm described in [1] cannot assure that the designed network is one-fault-tolerant. In fact, the crossover operator can create non fault-tolerant network from two networks that are one-fault-tolerant. The crossover operator and its function is the same as described in [1]. Algorithm for the crossover operator is as follows [1]:

1. Randomly choose two mating solutions $G(N, L_1)$ and $G(N, L_2)$ to crossover.
2. Randomly choose two links $l_{i,j} \in L_1$ and $l_{k,m} \in L_2$ such that: $l_{i,k} \notin L_1 \cup L_2$, $l_{i,m} \notin L_1 \cup L_2$, $l_{j,k} \notin L_1 \cup L_2$, $l_{j,m} \notin L_1 \cup L_2$.
3. Let: $\hat{L}_1 = L_1 \cup \{l_{k,m}\} - l_{i,j}$ and $\hat{L}_2 = L_2 \cup \{l_{i,j}\} - l_{k,m}$

If the $\hat{L}_1$ and $\hat{L}_2$ is not one-fault-tolerant according to $\deg(v) < 2$ launch the following repairing algorithm. Add necessary links from the following set. The links are chosen according to the cost of the link.
$\{l_{i,k}, l_{k,j}\}$:$\{l_{i,m}, l_{m,j}\}$:$\{l_{i,k}, l_{j,m}\}$:$\{l_{i,m}, l_{j,k}\}$

The resultant network should be one fault-tolerant according to [1]. In the following figure is possible to see that the crossover operator can create the network with the non fault-tolerant topology.

It is possible to see the function of the crossover operator and repairing algorithm in the Fig. 6.1 In the first step two networks are crossed. One of the results is in the second row at the left part. The network is not one-fault-tolerant and therefore the repairing algorithm was launched (links in the red colour).

The repaired network is again crossed with other network. One of the results is in the last row. It is possible to see that the resultant network is not one-fault-tolerant. If the link between nodes V_5, V_6 or V_6, V_7 is interrupted the network is divided into two parts that are not able to communicate between each other. It means that the network is not fault-tolerant and the algorithm described in [1] cannot ensure that the resultant network is fault-tolerant since the crossover operator can create non fault-tolerant network and the condition that should launch the repairing algorithm is not fulfilled since all nodes has $\deg(v) \geq 2$.

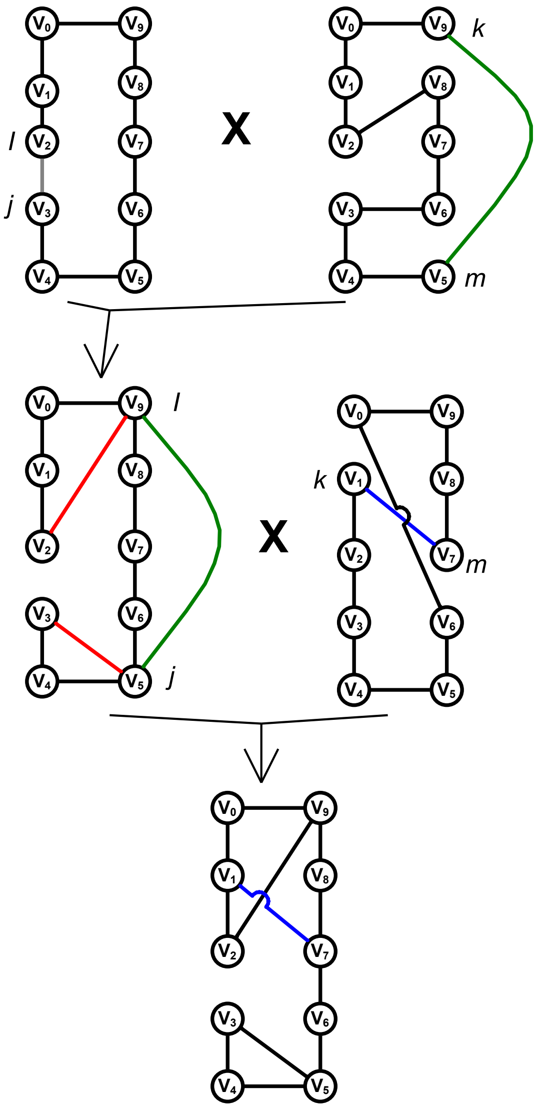

Figure 6.1: Crossover operator [1] - Proceeding

6.2 Setting of algorithm

These settings were used for examples described in this thesis.

6.2.0.1 Same fault-tolerance - literature

The setting of the algorithm for the same level of the fault-tolerance [2],[1], Tab. 4.1, Tab. 4.2; $p_m = 0.05$, $p_c = 0.25$, $N_p = 100$, $N_{ch} = 100$, $c_{i,j} = 100$ for $\forall i, j \in \{1, 2, .., N\}$, $m_{i,j} = k_F$, for $\forall i, j; i \neq j \wedge i, j \in \{1, 2, .., N\}$, $k_F \in \{3, 4, 5\}$

6.2.0.2 Same fault-tolerance - proposed algortihm

The setting of the algorithm for the node degree used as the fault-tolerance measure Tab. 4.3; $p_m = 0.05$, $p_c = 0.25$, $N_p = 100$, $N_{ch} = 100$, $c_{i,j} = 100$ for $\forall i, j \in \{1, 2, .., N\}$, $m_{i,j} = k_F$, for $\forall i, j; i \neq j \wedge i, j \in \{1, 2, .., N\}$, $k_F \in \{3\}$

6.2.0.3 Different fault-tolerance

The setting of the algorithm for the design of the physical topology with the different levels of the fault-tolerance in different parts of the network Tab. 4.4; $p_m = 0.05$, $p_c = 0.25$, $N_p = 100$, $N_{ch} = 100$, $c_{i,j} = 100$ for $\forall i, j \in \{1, 2, .., N\}$, $m_r = 3$, for $m_r \in \left\{m_{1,2}, m_{1,3}, .., m_{1,N/2}\right\}$

6.2.0.4 Logical topology design

The setting of the algorithm for the design of the logical topology Tab. 4.5. $p_m = 0.05$, $p_c = 0.02$, $N_{Lch} = 200$, $N_L = 200$, $cap = 1000000 bit/sec$, $Pack_{len} = 50 Bytes$ The matrix of the flows contained the elements that describes the situation in which every node receive and send approximately $10000 bits/sec.$. Matrix was created randomly for every number of nodes (Matrix of the maximal permitted delay had elements equal to 1). The algorithm was launched in order to find the time complexity of the algorithm.

6.2.0.5 Simulation results

The setting for the simulation results 4.6.2: Design of physical topology: $p_m = 0.05$, $p_c = 0.25$, $N_p = 100$, $N_{ch} = 200$. Logical topology: $p_m = 0.05$, $p_c = 0.25$, $N_{Lch} = 200$, $N_L = 200$, $cap = 100000 bit/sec$, $Pack_{len} = 50 Bytes$

$$F = \left\{ \begin{matrix} 0 & 1000 & 1300 & 1200 & 4000 & 0 & 1000 & 1500 \\ 4000 & 0 & 500 & 1000 & 500 & 2500 & 500 & 1000 \\ 3000 & 0 & 0 & 200 & 3000 & 3000 & 400 & 400 \\ 3000 & 700 & 4000 & 0 & 300 & 500 & 500 & 1000 \\ 1000 & 1500 & 3000 & 1500 & 0 & 1000 & 1800 & 200 \\ 500 & 500 & 1000 & 3000 & 1000 & 0 & 3000 & 1000 \\ 500 & 3000 & 1000 & 500 & 4000 & 0 & 0 & 1000 \\ 4000 & 1000 & 500 & 0 & 3000 & 1000 & 500 & 0 \end{matrix} \right\} Bytes$$

$$C = \left\{ \begin{matrix} 0 & 1 & 10 & 10 & 10 & 10 & 10 & 1 \\ 1 & 0 & 1 & 10 & 10 & 1 & 10 & 10 \\ 10 & 1 & 0 & 1 & 10 & 10 & 1 & 10 \\ 10 & 10 & 1 & 0 & 1 & 10 & 10 & 1 \\ 1 & 10 & 10 & 1 & 0 & 1 & 10 & 10 \\ 10 & 1 & 10 & 10 & 1 & 0 & 1 & 10 \\ 10 & 10 & 1 & 10 & 10 & 1 & 0 & 1 \\ 1 & 10 & 10 & 10 & 10 & 10 & 1 & 0 \end{matrix} \right\}$$

$$M = \left\{ \begin{matrix} 0 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 2 & 0 & 3 & 2 & 3 & 2 & 2 & 3 \\ 2 & 3 & 0 & 3 & 2 & 3 & 2 & 2 \\ 2 & 2 & 3 & 0 & 3 & 2 & 2 & 3 \\ 2 & 3 & 2 & 3 & 0 & 3 & 2 & 2 \\ 2 & 2 & 3 & 2 & 3 & 0 & 2 & 3 \\ 2 & 2 & 2 & 2 & 2 & 2 & 0 & 2 \\ 2 & 3 & 2 & 3 & 2 & 3 & 2 & 0 \end{matrix} \right\}$$

$$D = \begin{pmatrix} 0 & 0,1 & 0,55 & 0,25 & 0,4 & 0 & 0,15 & 0,55 \\ 0,6 & 0 & 0,55 & 0,2 & 0,5 & 0,3 & 0,2 & 0,1 \\ 0,6 & 0 & 0 & 0,1 & 0,3 & 0,6 & 0,6 & 0,45 \\ 0,25 & 0,55 & 0,3 & 0 & 0,3 & 0,3 & 0,35 & 0,3 \\ 0,35 & 0,2 & 0,5 & 0,1 & 0 & 0,45 & 0,3 & 0,2 \\ 0,4 & 0,45 & 0,35 & 0,1 & 0,5 & 0 & 0,8 & 0,25 \\ 0,5 & 0,2 & 0,3 & 0,2 & 0,6 & 0 & 0 & 0,75 \\ 0,4 & 0,3 & 0,45 & 0 & 0,4 & 0,4 & 0,15 & 0 \end{pmatrix}$$

$$D_{\mathrm{Res}} = \begin{pmatrix} 0 & 0,08 & 0,51 & 0,25 & 0,38 & 0 & 0,14 & 0,54 \\ 0,59 & 0 & 0,52 & 0,16 & 0,49 & 0,26 & 0,2 & 0,08 \\ 0,57 & 0 & 0 & 0,08 & 0,27 & 0,58 & 0,57 & 0,43 \\ 0,22 & 0,53 & 0,28 & 0 & 0,25 & 0,29 & 0,32 & 0,27 \\ 0,31 & 0,19 & 0,48 & 0,08 & 0 & 0,41 & 0,29 & 0,15 \\ 0,36 & 0,44 & 0,33 & 0,07 & 0,47 & 0 & 0,76 & 0,25 \\ 0,48 & 0,15 & 0,26 & 0,18 & 0,59 & 0 & 0 & 0,75 \\ 0,38 & 0,26 & 0,44 & 0 & 0,39 & 0,37 & 0,13 & 0 \end{pmatrix}$$

The comparison of the estimated behaviour and the simulation results corresponding to this example are in the directory: $..\backslash sim$.

6.2.0.6 Network expansion - 2-fault-tolerance

The setting for the network expansion - Tab. 4.7. The original network had a structure corresponding to the Fig. 4.27; $p_m = 0.2$, $p_c = 0.3$, $N_p = 100$, $N_{ch} = 100$; $c_{Fi,j} = 1$, $c_{Mi,j} = 10, c_{RMi,j} = 100, c_{RFi,j} = 100$, $m_{i,j} = 3$, $k_{mi} = 3$, $k_{fi} = 3$ for $\forall i, j \in \{1, 2, .., N\} \wedge i \neq j$, two nodes were added for every setting.

6.2.0.7 Network reduction - 2-fault-tolerance

The network reduction - Tab. 4.8. The original network had a structure corresponding to the Fig. 4.27; $p_m = 0.2$, $p_c = 0.3$, $N_p = 100$, $N_{ch} = 100$; $c_{Fi,j} = 1$, $c_{Mi,j} = 10, c_{RMi,j} = 100, c_{RFi,j} = 100$, $m_{i,j} = 3$, $k_{mi} = 3$, $k_{fi} = 3$ for $\forall i, j \in \{1, 2, .., N\} \wedge i \neq j$, $R_{ed} \in \{N-1, N\}$; two nodes: $N-1, N$ were removed for every setting.

6.2.0.8 Tree topology

The setting of the design of the tree topology Tab. 4.13; $p_m = 0.02$, $p_c = 0.3$, $N_p = 100$, $N_{ch} = 250$; $k_{mi} = 4$, $k_{fi} = 4$, $Depth_{max} = 4$ The cost matrices exactly describe the optimal topology that should be designed. Elements of the cost matrix for the metallic cable are 1000 excluding those elements that correspond to communication links in the following Fig. 6.2, Fig. 6.3, Fig. 6.4, Fig. 6.5, Fig. 6.6, Fig. 6.7. Elements that correspond to the communication links of Fig. 6.2, Fig. 6.3, Fig. 6.4, Fig. 6.5, Fig. 6.6, Fig. 6.7 are 1 for the metallic cable and 10 for the fibre cable.

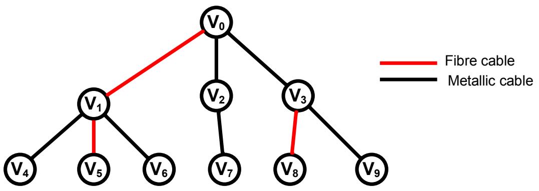

Figure 6.2: Costs - Tree topology $N = 10$

6.2.0.9 Network expansion - tree topology

The setting for the network tree topology expansion - Tab. 4.14; $p_m = 0.02$, $p_c = 0.3$, $N_p = 100$, $N_{ch} = 250$; $k_{mi} = 4$, $k_{fi} = 4$, $Depth_{max} = 4$, $c_{Fi,j} = 100$, $c_{RFi,j} = 100$, $c_{RMi,j} = 100$, $\forall i, j \in \{0, 1, 3, .., N-1\} \wedge i \neq j$, $c_{Mi,j} = 1$; for $j = 2$,

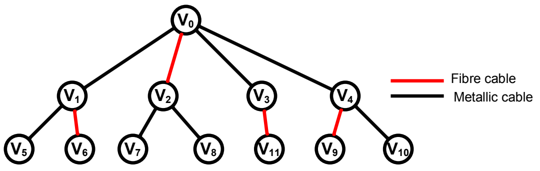

Figure 6.3: Costs - Tree topology $N = 12$

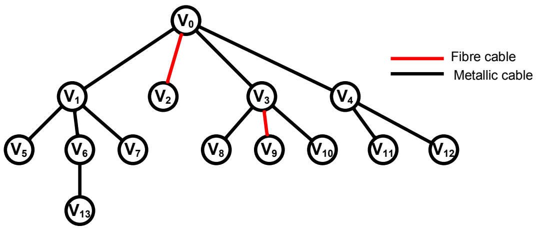

Figure 6.4: Costs - Tree topology $N = 14$

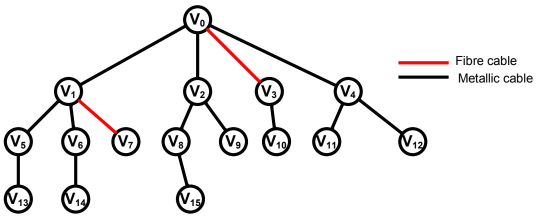

Figure 6.5: Costs - Tree topology $N = 16$

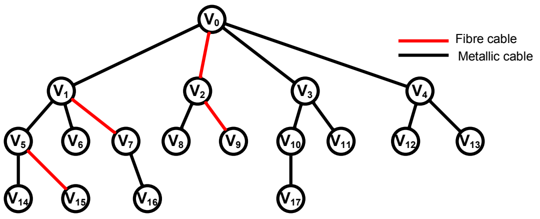

Figure 6.6: Costs - Tree topology $N = 18$

$c_{Mi,j} = 10$ for $\forall i, j \in \{0, 1, 3, .., N-1\} \wedge i \neq j \wedge j \neq 2$. The original network topology was the optimal topology gained by the design of the tree topology Fig. 6.2, Fig. 6.3, Fig. 6.4, Fig. 6.5, Fig. 6.6.

6.2.0.10 Network reduction - tree topology

The setting for the network tree topology reduction - Tab. 4.15; $c_{Fi,j} = 100$, $c_{RFi,j} = 100$, $c_{RMi,j} = 100$, $\forall i, j \in \{0, 1, 3, .., N-1\} \wedge i \neq j$, $c_{Mi,j} = 1$ for $j = 2$,

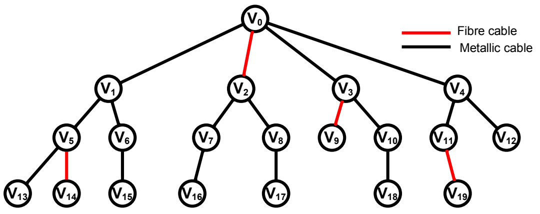

Figure 6.7: Costs - Tree topology $N = 20$

$c_{Mi,j} = 10$ for $\forall i, j \in \{0, 1, 3, .., N-1\} \wedge i \neq j \wedge j \neq 2$, $R_{ed} \in \{2\}$. The original network topology was the optimal topology gained by the design of the design of the tree topology Fig. 6.3, Fig. 6.4, Fig. 6.5, Fig. 6.6, Fig. 6.7.

6.2.0.11 Dependency of quality results on number of chromosomes

The designed network should have different levels of the fault-tolerance. The setting is the same as described in 6.2.0.3. The number of chromosomes was changed correspondingly to the figures. The probability of the finding of the optimal solution is the same as the probability for any other configuration. Therefore, it is possible to expect that there it is possible to get statistically the same quality of the results for the other demanded structure of nodes. For every setting was done minimally a hundred launches.

The curve ”min. same degree” is the best result that it is possible to reach when the algorithm that is able to design the network with the same level of the fault-tolerance in the whole network is used. The curve ”real-min” is the minimal cost of the network that exactly meets the demands for the fault-tolerance. The curve ”Genetic alg.” describes the results gained by the proposed algorithm that is able to design network with different levels of the fault-tolerance.

6.3 Contents of enclosed CD

List of directories

- Thesis - directory contains this thesis in .pdf format
- Sim - directory contains figures with comparison of the estimated behaviour and simulated behaviour of the network described in 6.2.0.5 and 4.6.2 . File routes.csv contains the names of data-flows and their communication path in the network depicted in the file net.jpg.

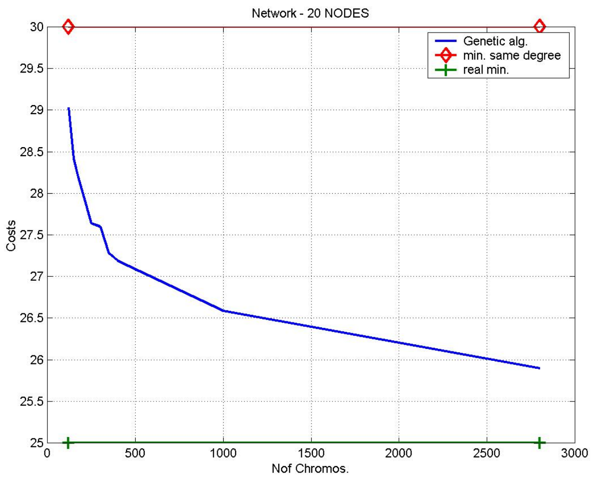

Figure 6.8: Costs - dependence on number of chromosomes $N = 20$

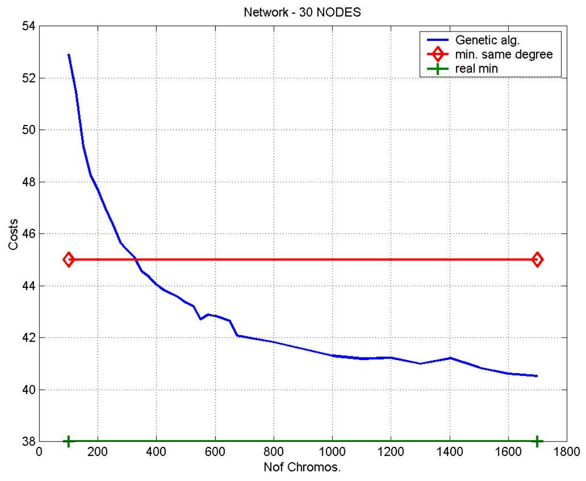

Figure 6.9: Costs - dependence on number of chromosomes $N = 30$

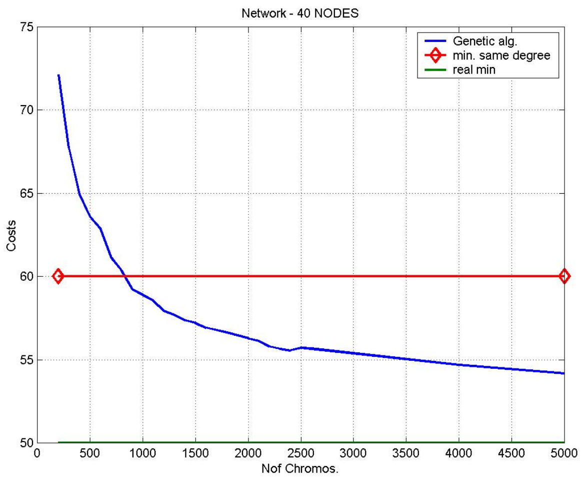

Figure 6.10: Costs - dependence on number of chromosomes $N = 40$

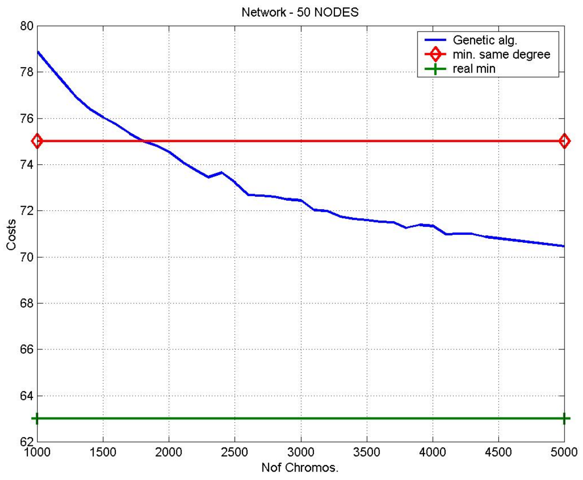

Figure 6.11: Costs - dependence on number of chromosomes $N = 50$

Bibliography

[1] S.-T. Cheng, “Topological optimization of a reliable communication network,” *IEEE Transactions on Reliability*, vol. 47, pp. 225 –233, sep 1998.

[2] E. Szlachcic, “Fault tolerant topological design for computer networks,” *International Conference on Dependability of Computer Systems, 2006. DepCos-RELCOMEX ’06.*, pp. 150 –159, may 2006.

[3] C.-S. Wang and C.-T. Chang, “Integrated genetic algorithm and goal programming for network topology design problem with multiple objectives and multiple criteria,” *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 680 –690, 2008.

[4] D. Zili, Y. Nenghai, and L. Zheng, “Designing fault tolerant networks topologies based on greedy algorithm,” *Third International Conference on Dependability of Computer Systems, 2008. DepCos-RELCOMEX ’08.*, pp. 227 –234, june 2008.

[5] N. Maxemchuk, I. Ouveysi, and M. Zukerman, “A quantitative measure for telecommunications networks topology design,” *IEEE/ACM Transactions on Networking*, vol. 13, no. 4, pp. 731 – 742, 2005.

[6] K.-T. Ko, K.-S. Tang, C.-Y. Chan, K.-F. Man, and S. Kwong, “Using genetic algorithms to design mesh networks,” *Computer*, vol. 30, pp. 56 –61, aug 1997.

[7] R. Elbaum and M. Sidi, “Topological design of local-area networks using genetic algorithms,” *IEEE Transactions on Networking*, vol. 4, no. 5, pp. 766 –778, 1996.

[8] B. Liu and K. Iwamura, “Topological optimization models for communication network with multiple reliability goals,” *Computers & Mathematics with Applications*, vol. 39, no. 7-8, pp. 59 – 69, 2000.

[9] E. G. Carrano, L. A. E. Soares, R. H. C. Takahashi, R. R. Saldanha, and O. M. Neto, “Electric distribution network multiobjective design using a problem-specific genetic algorithm,” *IEEE Transactions on Power Delivery*, vol. 21, pp. 995 –1006, April 2006.

[10] G. Kumar, N. Narang, and C. Ravikumar, "Efficient algorithms for delay-bounded minimum cost path problem in communication networks," *5th International Conference on High Performance Computing, 1998. HIPC '98.*, pp. 141 –146, dec 1998.

[11] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Cambridge, MA,USA: Springer Verlag, 1996.

[12] M. Mitchel, *An Introduction to Genetic Algorithms*. Cambridge, MA,USA: MIT Press Cambridge, 1999.

[13] J. H. Holland., *Adaptation in Natural and Artificial Systems*. Cambridge, MA,USA: MIT Press Cambridge, 1992.

[14] R. Diestel, *Graph Theory, electronic edition, third edition*. New York, 2005: Springer-Verlag Heidelberg, 2005.

[15] D. Reichelt, P. Gmilkowsky, and F. Rothlauf, "Designing reliable communication networks with a genetic algorithm using a repair heuristic," *Proceedings 4th European Conference, EvoCOP 2004,*, 2003.

[16] F. Altiparmak and B. Dengiz, "A cross entropy approach to design of reliable networks," *European Journal of Operational Research*, vol. 199, no. 2, pp. 542 – 552, 2009.

[17] R. Rubinstein, "The cross-entropy method for combinatorial and continuous optimization," *Methodology and Computing in Applied Probability*, vol. 1, no. 2, pp. 127–190, 1999.

[18] B. Rothfarb, H. Frank, D. M. Rosenbaum, K. Steiglitz, and D. J. Kleitman, "Optimal design of offshore natural-gas pipeline systems," *Operations Research*, vol. 18, no. 6, pp. 992 – 1020, 1970.

[19] M. Gerla and L. Kleinrock, "On the topological design of distributed computer networks," *IEEE Transactions on Communications*, vol. 25, no. 1, pp. 48 – 60, 1977.

[20] A. Dutta and S. Mitra, "Integrating heuristic knowledge and optimization models for communication network design," *IEEE Transactions on Knowledge and Data Engineering*, vol. 5, pp. 999 –1017, dec 1993.

[21] B. Dengiz, F. Altiparmak, and A. Smith, "Local search genetic algorithm for optimal design of reliable networks," *IEEE Transactions on Evolutionary Computation*, vol. 1, pp. 179 –188, sep 1997.

[22] R.-H. Jan, F.-J. Hwang, and S.-T. Chen, “Topological optimization of a communication network subject to a reliability constraint,” *IEEE Transactions on Reliability*, vol. 42, pp. 63 –70, mar 1993.

[23] A. Kumar, R.-M. Pathak, and Y.-P. Gupta, “Genetic-algorithm-based reliability optimization for computer network expansion,” *IEEE Transactions on Reliability*, vol. 44, pp. 63 –72, mar 1995.

[24] Saha and Chakraborty, “An efficient link enhancement strategy for computer networks using genetic algorithm,” *Computer Communications 20 (1997) 798-803*, 1997.

[25] D. Thompson and G. Gilbro, “Comparison of two swap heuristics with a genetic algorithm for the design of an atm network,” *Proceedings. 7th International Conference on Computer Communications and Networks, 1998.*, pp. 833 –837, oct 1998.

[26] Y. Chen, J. Li, and J. Chen, “A new algorithm for network probabilistic connectivity,” *Military Communications Conference Proceedings, 1999. IEEE MILCOM 1999.*, vol. 2, pp. 920 –923 vol.2, 1999.

[27] E. Szlachcic. and J. Mlynek, “Efficiency analysis in communication networks topology design,” *Fourth International Conference on Dependability of Computer Systems, 2009. DepCos-RELCOMEX ’09.*, pp. 184 –191, july 2009.

[28] J. Han, G. Malan, and F. Jahanian, “Fault-tolerant virtual private networks within an autonomous system,” *Proceedings of 21st IEEE Symposium on Reliable Distributed Systems, 2002.*, pp. 41 – 50, 2002.

[29] L. Berry, B. Murtagh, G. McMahon, S. Sugden, and L. Welling, “Genetic algorithms in the design of complex distribution networks,” *International Journal of Physical Distribution and Logistics Management*, vol. 28, 1998.

[30] T. Fencl, P. Burget, and J. Bilek, “Network topology design,” *Preprints of the 17th IFAC World Congress*, vol. 17, no. 1, 2008.

[31] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms, second edition.* New York, 2001: McGraw Hill New York, 2001.

[32] C. A. Coello, “An updated survey of ga-based multiobjective optimization techniques,” *ACM Comput. Surv.*, vol. 32, no. 2, pp. 109–143, 2000.

[33] L.-Y. Wang, J. Zhang, and H. Li, “An improved genetic algorithm for tsp,” *International Conference on Machine Learning and Cybernetics, 2007*, vol. 2, pp. 925 –928, aug 2007.

[34] Z. Hanzalek, P. Burget, and P. Sucha, “Profinet io irt message scheduling,” *Euromicro Conference on Real-Time Systems*, vol. 0, pp. 57–65, 2009.

[35] R. Ramaswami and K.-N. Sivarajan, “Design of logical topologies for wavelength-routed optical networks,” *IEEE Journal on Selected areas in communication*, vol. 14, pp. 840 – 852, June 1996.

[36] D.-N. Yang and W. Liao, “Design of light-tree based logical topologies for multicast streams in wavelength routed optical networks,” *Infocom 2003*, vol. 1, pp. 32 – 41, June 2003.

[37] N. Tsujii, K. Baba, and S. Tsukiyama, “An interconnect topology optimization by a tree transformation,” *ASP-DAC 2000*, vol. 1, pp. 93 – 98, June 2000.

[38] H.-A. Harutyunyan, C.-D. Morosan, and Y. Zhang, “Two tree-based algorithms for network spare capacity design,” *PCDAT 07*, pp. 279–284, Dec 2007.

[39] S.-C. Ku, B.-F. Wang, and T.-K. Hung, “Constructing edge-disjoint spanning trees in product networks,” *IEEE transaction on Parallel and distributed systems*, vol. 14, pp. 213 – 222, March 2003.

[40] D. England, B. Veeravalli, S. Member, and J.-B. Weissman, “A robust spanning tree topology for data collection and dissemination in distributed environments,” *IEEE Transactions on Parallel and distributed systems*, vol. 18, pp. 608 – 621, may. 2007.

[41] X.-Y. Li, Y. Wang, and W.-Z. Song, “Applications of k-local mst for topology control and broadcasting in wireless ad hoc networks,” *IEEE transaction on Parallel and distributed systems*, vol. 15, pp. 1057 – 1070, Dec. 2004.

[42] G. Xue and K. Thulasiraman, “Computing the shortest network under a fixed topology,” *IEEE Transactions on Computers*, vol. 51, pp. 499 – 501, sept. 2002.

[43] M. Pan, C. Chu, and P. Patra, “A novel performance-driven topology design algorithm,” *ASP-DAC, Proceedings of the 2007 Asia and South Pacific Design Automation*, vol. 2, pp. 244 – 249, aug 2007.

[44] J.-R. Kim, M. Gen, and K. Ida, “Bicriteria network design using a spanning tree-based genetic algorithm,” *Artifical life and robotics*, vol. 18, pp. 65 – 72, may. 1999.

[45] A. Juttner, A. Orban, and Z. Fiala, “Two new algorithms for umts access network topology design,” *European Journal of Operational Researche*, vol. 164, pp. 456 – 474, sep. 2005.

List of Publications

[Fencl et al.(2010a)Fencl, Burget, and Bílek] T. Fencl, P. Burget, and J. Bílek. Network topology design (**co-authorship 85% - paper in review**). *Control Engineering Practice*, 5 2010a. ISSN 0967-0661.

[Fencl et al.(2010b)Fencl, Burget, and Bílek] T. Fencl, P. Burget, and J. Bílek. Fast design of network topology (**co-authorship 90%**). In *The Second IFAC Symposium on Telematics Applications - Preprints*, pages 1–6, Bv. Republicii nr. 9, 300159 Timisoara, 2010b. Editura Politehnica.

[Fencl et al.(2008)Fencl, Burget, and Bílek] T. Fencl, P. Burget, and J. Bílek. Network topology design. In *Preprints of the 17th IFAC World Congress(**co-authorship 85%**)*, Seoul, 2008. IFAC. ISBN 978-3-902661-00-5.

[Fiala et al.(2008)Fiala, Burget, and Fencl] O. Fiala, P. Burget, and T. Fencl. Lablink - prostředí pro vzdálené laboratoře (**co-authorship 10%**). 2008. URL `lablink.felk.cvut.cz`.

[Burget et al.(2008)Burget, Fiala, Fencl, and Moc] P. Burget, O. Fiala, T. Fencl, and L. Moc. Remote labs and resource sharing in control systems education (**co-authorship 15%**). In *Preprints of the 17th IFAC World Congress*, Seoul, 2008. IFAC. ISBN 978-3-902661-00-5.

[Fencl and Bílek(2007a)] T. Fencl and J. Bílek. Network optimization. In *7th WSEAS International Conference on APPLIED INFORMATICS AND COMMUNICATIONS (**co-authorship 95%**)*, pages 92–97, Athens, 2007a. WSEAS. ISBN 978-960-8457-96-6.

[Fencl and Bílek(2007b)] T. Fencl and J. Bílek. Network optimization. In *Proceedings of the international Web conference CEEPUS Summer School 2007(**co-authorship 95%**)*, Maribor, 2007b. University of Maribor. ISBN 978-961-248-054-7.

[Fencl(2006a)] T. Fencl. Design of physical and logical topology of communication networks (**authorship 100%**). Technical report, CTU FEL DCE, Prague, 2006a.

[Fencl(2006b)] T. Fencl. Communication in the building technology (**authorship 100%**). In *Modern Trends in Control*, pages 73–82, Košice, 2006b. Equilibria. ISBN 80-969224-6-7.

[Fencl(2005)] T. Fencl. Modelling and control of heat exchanger station (**authorship 100%**). In *Intelligent Control Systems*, pages 22–28, Brno, 2005. University of Technology. ISBN 80-214-2976-3.

Vita

Tomáš Fencl received a degree in Electrical Engineering from the Czech Technical University (CTU) in Prague in 2005. Since 2005, he is a Ph.D. student at the same university. He was involved in research project euSophos and cooperates in project ALICE (Advanced Logic in Control Engineering). From 2006 to 2010, he worked as a research fellow with the Department of the Control Engineering.

His teaching activities included a broad scope of courses. He taught course Control systems and Logical systems. He was a leading teacher on Design of the Automated Systems and was responsible for the content of the laboratory exercises. He was also responsible for the content of the ALICE project where he was a leading teacher as well. Tomáš Fencl supervised bachelor and diploma theses as well as student projects under the CEPOT. He is an instructor in the Talnet project (teaching of the gifted youth). Talnet project is led by the Faculty of Mathematics and Physics of the Charles University and Department of Control Engineering cooperates at the project.

His research interests include networked and embedded systems, communication protocols and communication systems, his results were presented at international conferences. Tomáš also chaired the session (Session on remote sensor data acquisition) at the conference - IFAC TA2010. Currently, other paper [Fencl et al.(2010a)Fencl, Burget, and Bílek] is under review of magazine: Control Engineering Practice.