A Top-Down Heuristic for TCFA Problem in WAN

Roza Goscien, Iwona Pozniak-Koszalka, Leszek Koszalka, and Andrzej Kasprzak
Department of Systems and Computer Networks
Wroclaw University of Technology
Wroclaw, Poland
roza.goscien@gmail.com, {iwona.pozniak-koszalka, leszek.koszalka, andrzej.kasprzak}@pwr.wroc.pl

***Abstract*—This paper presents the results of investigation focused on a new topology design problem – a very important issue affecting nowadays Wide Area Networks (WANs). We formulate a T*CFA* (Topology, Capacity and Flow Assignment) problem for WANs and propose a novel heuristic algorithm to solve it. Moreover, we present findings of computational experiments, carried out to compare the properties of the created algorithm with other TCFA methods (both exact and heuristic) and also to determine dependences between processing time and dimensions of problems for TCFA tasks. The obtained results confirm that the proposed Top Down heuristic algorithm is promising.**

Keywords- algorithm; network design; optimization; WAN; TCFA

I. INTRODUCTION

The most important thing about Wide Area Network (WANs) is its necessity to connect Local Area Networks (LANs), PCs or terminals over large distances and to fulfill the users' requirements. Evidently, realization of abovementioned tasks is connected with some costs. Abovementioned requirements and bounded budget make modeling WAN so important and increasingly popular nowadays [1].

Modeling WAN includes two main groups of problems: (*i*) optimization of existing topologies, which are not efficient enough, (*ii*) new topologies design. These problems can be divided into three main groups: *FA* (Flow Assignment), *CFA* (Capacity and Flow Assignment), *TCFA* (Topology, Capacity and Flow Assignment) defined in detail in [1], and [2]. This paper is focused on T*CFA* problems.

In this work, the defined T*CFA* problem is considered. We propose our own algorithm to solve this problem. We present the results of the designed computational experiments, carried out in order to compare the proposed algorithm with other algorithms solving TCFA (both exact and heuristic) and to determine dependences between processing time and dimensions of problems to be solved as well.

Optimization problems like FA, CFA and TCFA were a subject of many papers. TCFA problems are the most complex. Different forms of TCFA problems were discussed and examined in literature. Walkowiak [3] considered the problem with no network topology given a priori – placement of networks' nodes was a part of the optimization task. The problem objective was system's overall costs. In the same paper, various methods solving problem were presented – for example, algorithm based on Lagrangian relaxation. Other forms of TCFA problem were presented by M. Gerla and L. Kleinrock [4], where the authors considered several propositions for problem objective and constraints formulation. The notation of TCFA problem used in this paper was presented by A. Kasprzak [5], where a heuristic algorithm to solve the problem was presented. We used this algorithm to evaluate our own algorithm.

The paper is organized as follows. In Section II, we formulate the formal model of the considered problem. Section III presents our novel *Top-Down* algorithm. Findings from computational experiments are presented in Section IV. The final remarks and conclusion appear in Section V.

II. OPTIMIZATION PROBLEM

A. *The model of a computer network*

The model of a computer network can be created using graph theory. Graph's vertices $V=\{v_1, v_2, ..., v_n\}$ correspond to network's nodes and graph's edges $E=\{e_1, e_2, ..., e_m\}$ correspond to links. Numbers $c(e)$ assigned to edges determine links' capacities (in, e.g., bits per second) [6]. We considered only directive graphs in modeling.

In computer memory, a model can be saved as a weighing matrix N or adjacency matrix N' and vector of capacity c. There is an example of computer network model presented in Fig. 1 and described as below.

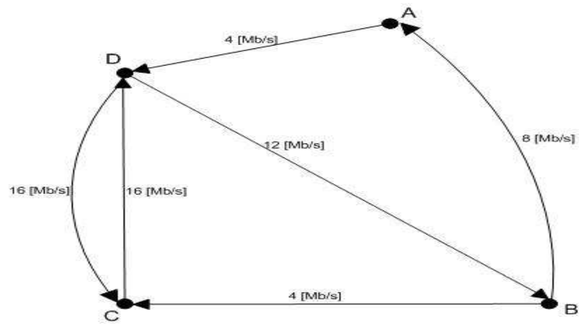

Figure 1. An example of computer network model.

$$N = \begin{bmatrix} 0 & 0 & 0 & 4 \\ 8 & 0 & 4 & 0 \\ 0 & 0 & 0 & 16 \\ 0 & 12 & 16 & 0 \end{bmatrix}, N' = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}, c = [4 \quad 8 \quad 4 \quad 16 \quad 12 \quad 16]$$

Weighting matrix $N=[n_{ij}]_{nxn}$ contains capacities of all existing links. n_{ij} is equal to 0 if the link between nodes i and j does not exist. Otherwise, there is a capacity value of this link. Adjacency matrix $N'=[n_{ij}]_{nxn}$ informs only whether a specific link exists. n'_{ij} is equal to 0 if the link between nodes i and j does not exist. Otherwise, there is a value 1. Links' capacities are saved sequentially in vector c.

B. *Multicommodity flows*

Multicommodity flow in WAN is defined as the average flow of information in a particular slot of time. Flow commodity is a set of packets with the same *i-th* source and *j-th* destination node.

Let r_{ij} denotes the average packet rate transmitted from node i to node j and $R=[r_{ij}]_{nxn}$ is a demand matrix, defined in [5] and [7]. Note that *d-th* flow commodity is connected with a pair of nodes: source s_d and termination t_d. The value $h_d=r_{ij}$, where $i=s_d$ and $j=t_d$, is known as volume of *d-th* commodity.

Let $x=e^-$ and $y=e^+$ be a source and destination node of link e, respectively.

Mathematically, multicommodity flow is a function $f^k:E\to R^+\cup 0;\ k=1,2,...,q$, which assigns to networks links $e\in E$ values $f^k(e)$ [b/s] satisfying constraints (1) – (3) (see [2] and [8]).

- for v∈ V and k=1,2, ..., q:

$$\sum_{e^+=v} f^k(e) - \sum_{e^-=v} f^k(e) = \begin{cases} r_k, v=s_k \\ -r_k, v=t_k \\ 0, v\neq s_k \wedge v\neq t_k \end{cases} \tag{1}$$

- for e∈ E and k=1,2, ..., q:

$$f^k(e)\geq 0 \tag{2}$$

$$f(e)=\sum_{k=1}^{q} f^k(e)\leq c(e) \tag{3}$$

The value *f(e) [b/s]* is known as an entire link's flow. Abovementioned formulation is based on node-link notation of the flow. We considered bifurcated flows as defined in [8],[9].

C. *Optimization criteria*

In optimization tasks, we used two mathematical dependences: cost of network and average packet delay.

The cost of a network is a sum of leasing all networks' links with specified capacities. Formally, it is described by formula (4). This goal function was used only as an additional constraint in T*CFA* problems.

$$d(\mathbf{e})=\sum_{e=(x,y)\in E} k_{e,c} \tag{4}$$

where $k_{e,c}$ is a cost of leasing link e with capacity $c=c(e)$.

The goal function in optimization problems was the average packet delay – nonlinear function of flow, described by formula (5).

$$T(\mathbf{f})=\frac{1}{\gamma}\sum_{e\in E}\frac{f(e)}{c(e)-f(e)} \tag{5}$$

where $\gamma=\sum_{k=1}^{q} r_k$ represents the average packet rate transmitted in the network per second.

D. *TCFA formulation*

We follow the formulation of TCFA problem presented e.g., in [9].

Given:

- Location of network's nodes,
- Demand matrix *R*,
- Set of feasible links,
- Sets of candidate links' capacities and corresponding cost of leasing – matrixes *C* and *D* respectively,
- *costBound* – as a maximum acceptable cost of topology,
- average packet size in the network in b/s - *pSize*.

Minimize:

- Average packet delay.

Over:

- Bifurcated flows,
- Links' capacities,
- Network's structure.

Subject to constraints:

- Realization of demand matrix,
- For each link e: link's flow does not exceed link's capacity,
- Sum of leasing all selected links does not exceed *costBound*.

III. TOP DOWN ALGORITHM

In this section, we present the *Top-Down* algorithm for TCFA problem in WAN. The flow chart of this method is presented in Fig. 2.

The idea of the algorithm is based on starting analyzing the problem with fully adjacency matrix *N'(i=0)* and gradually removing useless links or least used links, until the topology satisfies constraint *costBound* and enables to fulfill all traffic demands.

The choice of links for removing from topology is made using criterion δ(*e*) formulated as follows:

$$\delta(e) = \frac{f(e)}{c(e) - f(e)} \qquad (6)$$

Where *c(e)* is a capacity of link *e* and *f(e)* is the flow allocated to this link.

Criterion δ(*e*) is a ratio of flow allocated to link *e* to free (still available) capacity of this link. The smaller the criterion *δ(e,)* the higher the probability that link *e* will be removed from the topology.

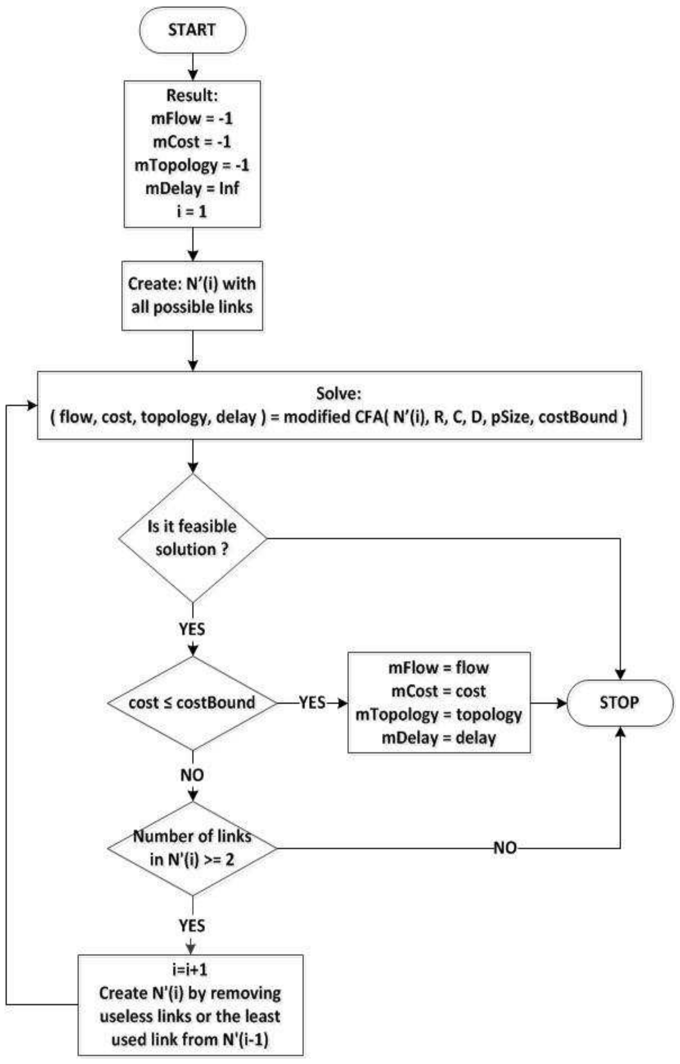
START
Result:
mFlow = -1
mCost = -1
mTopology = -1
mDelay = Inf
i = 1
Create: N'(i) with all possible links
Solve:
(flow, cost, topology, delay) = modified CFA(N'(i), R, C, D, pSize, costBound)
Is it feasible solution ?
YES
cost ≤ costBound
YES
mFlow = flow
mCost = cost
mTopology = topology
mDelay = delay
STOP
NO
Number of links in N'(i) >= 2
NO
YES
i=i+1
Create N'(i) by removing useless links or the least used link from N'(i-1)

Figure 2. The flow chart of *Top-Down* algorithm.

In each iteration, the *modified-CFA* method runs. This method finds a feasible solution that satisfies all current problems' constraints or solution that exceeds *costBound* of the smallest value.

The potential final solution is marked with "*m*" prefix, e.g., *mCost* is a cost of final topology.

IV. INVESTIGATION

In this section, we present and discuss the results of computational experiments. The goal of experiments was twofold. First, we tried to determine the dependence between processing time of TCFA algorithms d the dimension of the input data (size of optimization problem).

Second, we wanted to evaluate the proposed *Top-Down* algorithm, based on comparison with other methods. We considered the following comparison criteria: processing time, number of returned optimal and feasible solutions.

Top-Down algorithm was compared with two different TCFA methods: exact algorithm based on complete search and heuristic proposed by M. Gola [9], called as *Modified topology* in this paper. Abovementioned methods can run with different CFA algorithms. In our work we used exact CFA algorithm based on complete search and heuristic *Bottom-Up,* proposed by R. Goscien [6]. Thus, we examined five different TFCA algorithms.

The input data for algorithms were sets of random, solvable topologies, regardless of *costBound*. All presented results are averages of 4 – 10 measurements. We used solutions returned by exact TCFA algorithm as a reference, optimal solution.

Because of unreasonable processing time of exact algorithm, we compared only results for the number of 2 and 3 network nodes. Thus, the presented investigation is only a first step in evaluating our *Top-Down* algorithm and its results should help to decide if further work with greater and real topologies (e.g., 10 nodes) is promising.

To solve partial FA problems, a nonlinear programming technique was used.

A. *Impact of nodes' number on processing time*

We examined topologies with 2 flow commodities and 2 candidate links' capacities. We assumed *costBound* as a 2/3 of sum of the most expensive candidate capacities for all links.

We compared only results for the number of 2 and 3 network nodes. The results are presented in Table I.

TABLE I. PROCESSING TIME AS A FUNCTION OF NODES' NUMBER (SECONDS)

TCFA method + CFA method	**Number of nodes**	
	2	***3***
Exact TCFA + exact CFA	2,003	271,901
Exact TCFA + Bottom-Up	0,374	26,458
Modified topology + exact CFA	0,807	902,476
Modified topology + Bottom-Up	0,282	1,107
Top-Down	0,118	31,706

The processing time of all methods increases with the number of nodes. The explanation is based on some proportions. The number of nodes in the network determines the maximum possible number of links and flow commodities in topology. Moreover, dimensions of constraints matrixes used in nonlinear programming techniques are proportional to the number of nodes. The explanation of this fact was discussed in [6].

In this case we wanted to emphasize that the number of nodes in the topology significantly influences processing time, particularly in case of exact method, and to show that even for relatively small topologies processing time of exact method can increase to unacceptable value.

B. Impact of candidate capacities' number on processing time

We examined topologies with 3 nodes and 2 flow commodities. We assumed *costBound* as a 2/3 of sum of the most expensive candidate capacities for all links. The results are presented in Table II.

TABLE II. PROCESSING TIME AS A FUNTION OF CANDIDATE CAPACITIES' NUMBER (SECONDS)

TCFA method + CFA method	**candidate capacities' number**	
	2	***3***
Exact TCFA + exact CFA	131,662	1847,504
Exact TCFA + Bottom-Up	17,903	19,635
Modified topology + exact CFA	309,178	2696,440
Modified topology + Bottom-Up	0,924	1,002
Top-Down	2,976	27,211

According to our findings, we can say that processing time of TCFA methods increases with candidate capacities' number. Moreover, this influence is very significant for both – exact and heuristic algorithms.

C. Impact of flow commodities' number on processing time

In this case, we used input data with 3 nodes, and 2 candidate capacities and *costBound* assumed as in previous point.

TABLE III. PROCESSING TIME AS A FUNCTION OF FLOW COMMODITIES' NUMBER (SECONDS)

TCFA method + CFA method	**Flow commodities' number**			
	1	***2***	***3***	***4***
Exact TCFA + exact CFA	53,382	124,106	178,364	176,292
Exact TCFA + Bottom-Up	7,774	37,405	43,837	45,033
Modified topology + exact CFA	63,730	467,398	564,145	479,912
Modified topology+Bottom-Up	0,287	3,704	3,795	11,577
Top-Down	0,158	2,943	2,923	64,190

The results proved that the number of flow commodities influences processing time of TCFA algorithms. Moreover, according to the findings presented in Tables 1-3, we can say that *Top-Down* is the third best method in terms of processing time.

D. Impact of costBound constraint on processing time

In this part of our work, we discuss changes of processing time as a repercussion of different value of constraint *costBound*.

The results are presented for a topology with 3 nodes, 3 flow commodities and 2 candidate links' capacities.

As we can see in Fig. 3, methods based on exact CFA solve problems in the longest time. Furthermore, processing time of this algorithms increases very fast with constraint *costBound* – this is a consequence of enlarging space of topologies that we can create and analyse in CFA tasks.

After the cost of the most expensive topology is reached (about 710 euro) the processing time is approximately steady in relation to *costBound*.

Both figures (Fig. 3 and Fig. 4) also present one more important issue. The *Top-Down* algorithm is a method that solves problems in relatively short and acceptable time.

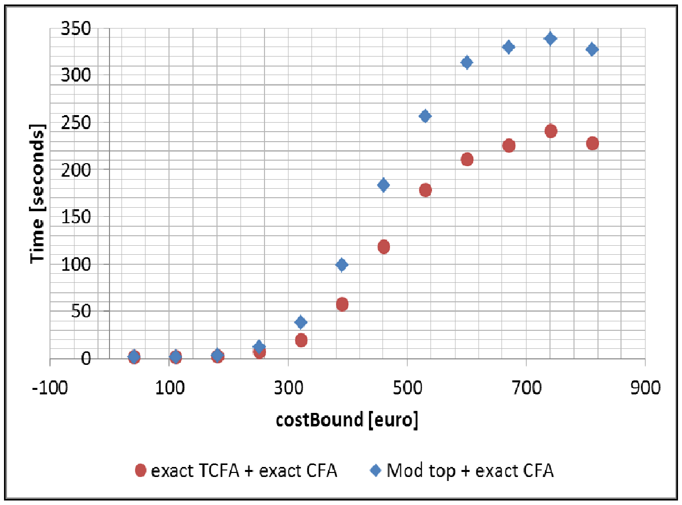

Figure 3. Processing time in relation to *costBound* – methods based on exact CFA.

In Fig.4, the results for the fastest algorithms are presented. The connection of exact TCFA and CFA heuritic *Bottom-Up* solves problems in the longest time, regardles of *costBound*. *Top-Down* method works faster.

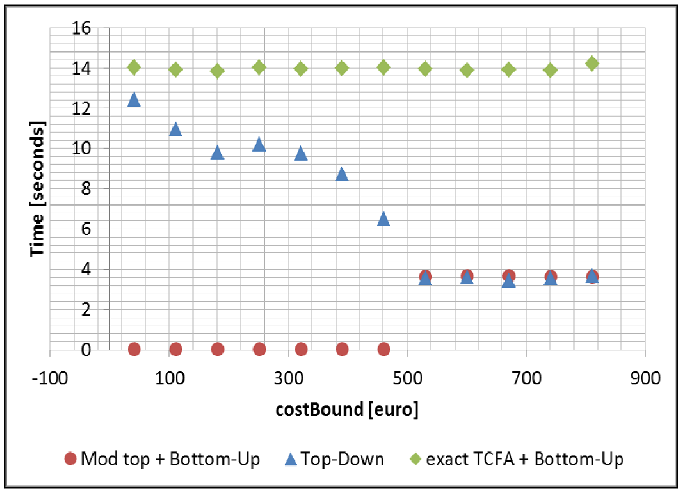

Figure 4. Processing time in relation to *costBound* – heuristic methods.

The processing time of *Top-Down* is the longest at the beginning, when *costBound* is small and we iteratively try to find a topology that is cheaper that our budget limit. Processing time of *Top-Down* algorithm descreses with incresing *costBound*, until the cost of one of the most expensive topologies is reached. Since changes of *costBound* do not influence on processing time – the space of potential solutions does not enlarge significantly.

When analysing dependences between dimensions of problems and processing time of solving methods, it is important to notice that:

- Values of all input data influence on processing time of TCFA algorithms,
- Processing time of exact algorithm is always longer than processing time of heuristic algorithm for the same set of input data,
- Exact TCFA algorithm is not always able to solve problem in reasonable time.

E. Comparison of algorithms

Overall, we compared 5 different TCFA algoritms, both exact and heuristic.

In terms of processing time, the most attractive are methods based on heuristic CFA and the proposed *Top-Down* method.

The percentage results of comparison according to number of returned optimal and feasible solutions are presented in Fig. 5 and Fig. 6. The solutions returned by connection of exact TCFA and exact CFA methods were used as reference.

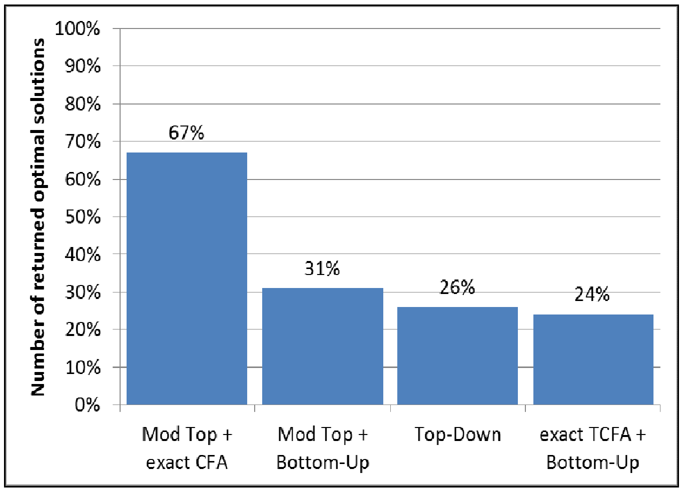

Figure 5. Number of returned optimal solutions.

The efficiency of TCFA methods that can run with different CFA methods (e.g. presented exact TCFA and *Modified Topology*) depends on the efficiency of the used CFA algorithm.

In terms of the number of the returned optimal solutions, the best was the method based on exact CFA – this method returned optimal solution in 67% of all expirements. The probalily of reaching optimal solution by *Top-Down* algorithm is smaller.

The result achieved by *Top-Down* was of 26%, but it is important to notice that processing time of *Top-Down* is shorter that processing time of methods based on exact CFA.

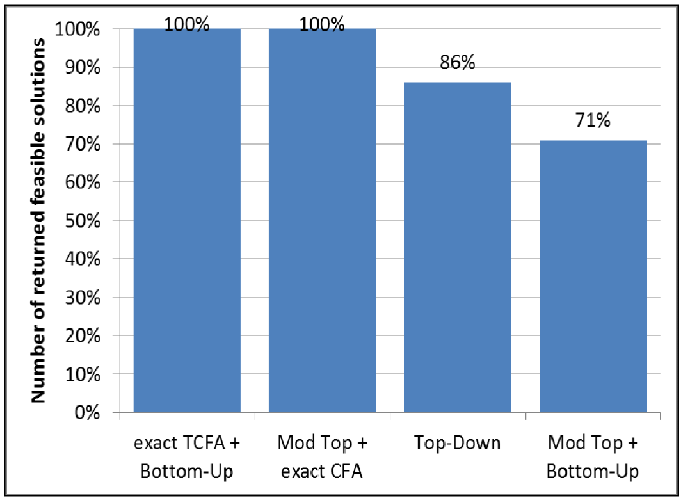

Figure 6. Number of returned feasible solutions.

The number of returned feasible solutions was equal to 100% for methods based on exact CFA or exact TCFA approach. For typically heuristic algorithms (*Top-Down*, connection of *Modified topology* and *Bottom-Up*), results were fewer. This is very important to underline in this comparison, that *Top-Down* returned more feasible solutions for solvable topologies than second heuristic method.

To summarize the evaluation of the*Top-Down* algorithm, it is important to notice that

- It is a method with a high number of returned feasible solutions and short, acceptable processing time, especially in comparison with other methods.
- If optimal solution is a crucial issue – the exact method has to be used.
- When some deviation from optimal solution is acceptable but there is restrictive time constraint then the heuristic method should be applied and in that case *Top-Down* algorithm is a good, candidate tool.

V. CONCLUSION

In the summary of all computational experiments, it is important to emphasize, that TCFA optimization problems are tasks with high computational and memory complexity (especially when the goal function is nonlinear), even for relatively small computer networks.

Moreover, complexity of these problems increases with increasing number of dimensions of input data. The growth of the problem complexity is connected with greater demands for memory and time, which are necessary to find optimal solution by exact methods.

Because of time and technical constraints, there is a necessity to find ways to solve optimization tasks using fewer resources.

This is the main reason and purpose of inventing heuristic methods – algorithms, which can find feasible solutions using fewer resources (e.g., time, memory) than exact method.

The selection of a suitable algorithm is a compromise between the accuracy of the solution and the required resources to solve the problem. Depending on specified requirements, another algorithm may be optimal tool. To choose the best one, some factors to be considered are: computer/technical equipment (environment), time constraint, allowed deviation from optimal solution.

In the further research in the area, the authors are planning to consider the algorithms based partially on the evolutionary approaches, e.g., presented in [10].

There are also several interesting issues that might be considered in the future work on this project. The most important include experiments with: more exact/heuristic algorithms, path-link notation of flow, larger topologies (number of nodes greater than 5) and detailed analysis of computational complexity of algorithms and memory usage following the ideas of multistage experiment design presented in [11].

ACKNOWLEDGMENT

This work was supported by the statutory funds of the Department of Systems and Computer Networks, Faculty of Electronics, Wroclaw University of Technology, No S20010W4.

REFERENCES

[1] T. Miksa, L.. Koszalka, and A. Kasprzak,. “Comparison of heuristic methods applied to optimization of computer networks”, Proceedings of the 11-th International Conference on Networks, ICN 2012, IARIA, cop. 2012., pp. 34-38.

[2] K. Walkowiak, “Ant algorithm for flow assignment in connection-oriented networks”, Int. J. Appl. Math. Comput. Sci.,vol. 15, No. 2, 2005, pp. 205–220.

[3] B. Gavish, “Topological design of computer communication networks”, Proceedings of the 22th Annual Hawaii International Conference on System Sciences, vol. III , 1989, pp. 770-779

[4] M. Gerla and L. Kleinrock, “On the topological design of distributed computer networks”, IEEE Transactions on Communications, vol. COM-25, 1977, pp. 48-60

[5] A. Kasprzak, Wide Area Networks, OWPW Publishing House, Wroclaw, 2001 /in Polish//.

[6] R. Goscien, W. Kmiecik, P. Ryba, I. Pozniak-Koszalka, and A. Kasprzak, “Bottom-Up heuristic for CFA problem in WAN”, Proceedings of International Conference on Systems Engineering 2012, pp. 85-90

[7] M. Pioro and D. Medhi, Routing, Flow and Capacity Design in Communication and Computer Networks, Morgan Kaufman Publishers, San Francisco, 2004.

[8] K. Walkowiak, “Heuristic algorithms for assignment of non-bifurcated multicommodity flows”, Proceedings of the XXVth International Autumn Colloquium, MARQ, 2003,. pp. 243-248.

[9] M. Gola and A.. Kasprzak, “Topology design problem with combined cost criterion and time varying traffic in wild area networks”, Proceedings to 17-th IMACS World Congress, Ecole Centralle de Lille, vol. 5, 2005.

[10] D. Ohia, L. Koszalka, and A. Kasprzak, “Evolutionary algorithm for solving congestion problem in computer network”, Lecture Notes in Computer Science, Springer, vol. 5711, 2009, pp. 112-121.

[11] L. Koszalka, D. Lisowski, and I. Pozniak-Koszalka, “Comparison of allocation algorithms for mesh structured networks using multistage simulation”, Lecture Notes in Computer Science, Springer, vol. 3984, 2006, pp.58-67.