Comparison of Heuristic Methods Applied to Optimization of Computer Networks

Tomasz Miksa, Leszek Koszalka, Andrzej Kasprzak
Department of Systems and Computer Networks,
Wroclaw University of Technology,
Wroclaw, Poland
e-mails: tom.miksa@gmail.com, leszek.koszalka@pwr.wroc.pl, andrzej.kssprzak@pwr.wroc.pl

***Abstract*—This paper presents an attempt to solve Capacity and Flow Assignment (CFA) problem, which is NP-complete. Meta-heuristic and heuristic algorithms are invented in order to find not only feasible but also effective solution. A set of test network instances, with provided dual bounds as a reference, is used to: tune algorithms' parameters, conduct experiments and assess results. Final results provide statistical measures derived from experiments and imply which of proposed algorithms provides better solutions. However, the two created algorithms seem to be promising.**

Keywords-computer networks; heuristic algorithm; simulated annealing; flow assignment; simulation

I. INTRODUCTION

The problems connected with design of Wide Area Networks (WAN) are important due to its practical application. Slight difference in solution quality has a big impact on networks maintenance cost. The issues of network design can be grouped in three categories: Flow Assignment (FA), Capacity and Flow Assignment (CFA), and Topology, Capacity and Flow Assignment (TCFA). Different optimization criteria are considered, but in most cases cost and average packet delay are chosen.

This paper deals with CFA for WAN problem with a cost as a criterion. Demands for transfer of data between multiple nodes may change during lifetime of already designed network. Then, in order to reduce upkeep cost, an optimization of routing paths (flow represents routing) and capacity modules installed on links should be made. Furthermore, solution of CFA problem can be used by algorithms that solves more complex problem – TCFA problem [1] [2].

CFA problem is NP-complete [3]. An algorithm of branch and bound was proposed in [1]. Authors of papers [4] [5] [6] applied successfully heuristic approaches. Other heuristic approaches are discussed in [7]. Some strongly polynomial algorithms are described in [8]. Meta-heuristic algorithm MSA and heuristic algorithm MHU are proposed in this paper, in order to solve CFA problem. Both algorithms are original implementation of heuristic approaches described in [1] [9] [10]. Some clues from [11] concerning integer programming are used. What is more, an experimentation system Ultimate Capacity and Flow Assignment (UCFA) was designed and implemented in order to test the proposed algorithms. This system enables to compare results provided by MSA and MHU algorithms against dual bounds or optimum solutions. This is achieved by the use of *sndlib* test instances [12].

The paper is organized as follows. In Section II, problem formulation is presented. The considered algorithms for solving such a problem are described in Section III. In Section IV, neighborhood types that are used by algorithms described in the previous section are presented. In Section V the designed and implemented experimentation system is described. The results of investigations are presented and discussed in Section VI. Finally, conclusions and prospects for future are presented in Section VII.

II. PROBLEM STATEMENT

The CFA problem consists in finding such a multi-commodity flow and capacity modules allocation that satisfies conditions arising from network topology, traffic matrix, etc.

In this paper, the CFA problem is formulated as follows, using notation from [12].

Constants:
D – number of demands,
E – number of edges,
V – number of vertices.

Indices:
$d = 1, 2, \ldots, D$ – demands,
$e = 1, 2, \ldots, E$ – edges,
$v = 1, 2, \ldots, V$ – vertices,
$k = 1, 2, \ldots, K$ – capacity module type,
$p = 1, 2, \ldots, P_d$ – paths for demand.

Indexed constants:
P_d – number of paths for demand d,
h_d – value of demand d,
c_e – capacity of edge e,
M_k – size of the link capacity module of type k,
δ_{edp} - equals 1, if link e belongs to path p of demand d, equals 0, otherwise,
ξ_{ek} - cost of module type k for edge e.

Variables:
x_{dp} – flow allocation vector.

Objective:

$$\min F = \sum_e \sum_k \xi_{ek} \tag{1}$$

Constraints:

$$\sum_p x_{dp} = h_d, \qquad d = 1,2,\ldots,D, \tag{2}$$

$$\sum_d \sum_p \delta_{edp} x_{dp} \le \sum_k M_k, e = 1,2,\ldots,E. \tag{3}$$

III. Algorithms

A. *MSA algorithm*

MSA algorithm is based on meta-heuristic approach known as Simulated Annealing (SA). SA algorithm imitates process of annealing applied in metallurgy. A description of SA can be found in [13]. Key factors for SA algorithm are the way the "neighborhood" is represented and the way the "moves" are made. These factors are closely related to the problem which is to be solved by SA. "Moves", which make a significant difference between "neighbors", should be made in high temperatures. However, when the temperature is close to end temperature, "moves" should be slight. This is the adaptive divide-and-conquer effect [10].

MSA algorithm uses two types of neighborhood:

- Single Random Any Capacity (SRAC),
- Single Random Decrease Capacity (SRDC).

The former is used as default, while the latter one is used when the temperature of current iteration is close to end temperature. SRAC and SRDC are described below.

MSA input parameters are:

- number of possible paths for demand – *KPath*,
- start temperature T_s,
- end temperature T_k,
- temperature interval τ,
- iterations number L.

Best values for above parameters are derived from experiments.

B. *MHU algorithm*

MHU algorithm is based on a concept presented in [1], which suggests that neighborhood solutions of CFA problem should be browsed in directed way. Direction should be the "excess of unused capacity". In other words, in following iterations, links chosen for modification are not randomly chosen like in MSA algorithm, but according to the amount of unused capacity. Such an approach results in fast increase of solution quality. Kasprzak [1] claims that such approach can lead to optimum solutions.

MHU algorithm starts its operation by installing maximum capacity modules on links. Multi-commodity flow is found, and objective is calculated. If it is not possible to find multi-commodity, the problem is unsolvable. Else, the link that has the biggest excess of unused capacity is chosen. Bandwidth of this link is reduced to previous capacity module from increasing sequence of available capacity modules for given link. If capacity module already installed on this link cannot be decreased (is already first in sequence), next link with biggest excess of unused capacity is chosen. If none of links can be modified, algorithm stops. Once the capacity module installed on chosen link has been modified, multi-commodity flow is calculated. In this way, new neighbor solution is generated. The step described above is repeated given number of times (algorithm's input parameter), or when there are no more links available for modification. In any iteration, solution with best objective is chosen. The result is used as a base solution for the next iteration. In the case when the solution generated in one of iterations is not feasible, because it is not possible to find multi-commodity flow, algorithm proceeds using best, last known, feasible solution.

IV. Neighborhood

This section clarifies the concept of neighborhood used for solving CFA problem. Neighbor solution is a new solution that is, in some way, modified when compared to the previous one. In CFA, the problem it consists of:

- link configuration vector.
- demand routing vector.

Link configuration vector consists of elements representing capacity modules installed on following links. Routing of demands is also represented by vector. Following elements are indexes of paths available for given demand. Set of available paths for each demand is created once, before neighboring solution is generated. In order to create this set algorithms like Dijkstra's, Bellman- Ford's or K-ShortestPath's algorithm can be used. Quantity of paths found for each demand can be done arbitrary or empirically.

A. *Single Random Any Capacity (SRAC)*

In this method of obtaining neighboring solution, operations are made on both link configuration and demand routing vectors. Index of link, that will be modified, is drawn. Then, an index from the list of available capacity modules is drawn. Drawn capacity module is inserted into vector replacing previously installed capacity module in according to drawn link index. For example, in Fig. 1 link that index is 5 was drawn. Capacity module installed on this link in base solution is 128kbps. Neighbor solution has on that link capacity module whose bandwidth is 512 kbps, because index 8 was drawn as a pointer for a list of capacity modules.

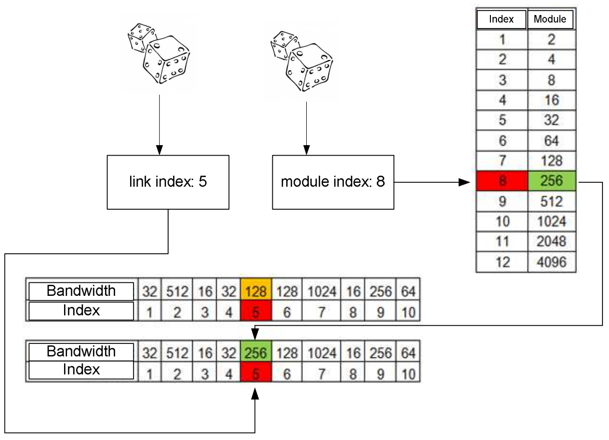
Index
Module
1 2
2 4
3 8
4 16
5 32
6 64
7 128
8 256
9 512
10 1024
11 2048
12 4096
link index: 5
module index: 8
Bandwidth 32 512 16 32 128 128 1024 16 256 64
Index 1 2 3 4 5 6 7 8 9 10
Bandwidth 32 512 16 32 256 128 1024 16 256 64
Index 1 2 3 4 5 6 7 8 9 10

Figure 1. SRAC neighborhood generation scheme.

Other links are left unmodified, according to base solution. Demand routing vector is created anew. For each demand, path's index from list of available paths is drawn.

B. *Single Random Decrease Capacity*

This method of creation of new neighboring solution is similar to SRAC method, described in previous section. Main difference lies in the way capacity module of drawn link is changed. Instead of random change of capacity module, lower capacity module is used, on condition it exists. For example if drawn link index is 6, only this link is modified. If capacity module installed on that link was 64kbps, and the lower capacity module is 32kbps, then 32kbps capacity module is installed. Demand routing vector is created anew, like it is in SRAC method.

V. EXPERMIENTATION SYSTEM

Experimentation system called Ultimate Capacity and Flow Assignment (UCFA) was created in order to solve CFA problem with a use of MHU and MSA algorithms. System possesses user friendly Graphical User Interface (GUI), which facilitates experimentation. UCFA enables among other things following options:

- Choice of test instances.
- Choice of algorithm.
- Setup of input parameters.
- Multithreaded tests.
- Live progress preview.
- Solution save.
- Results summary save.

Test instances are delivered with the experimentation system. Every instance possesses scheme reflecting nodes and links allocation. All instances are imported from *sndlib* library [12]. This guarantees that instances have always not only at least one feasible solution, but also provided test data is derived from real systems, what makes the simulations more realistic. Furthermore, information about network and problem parameters are available, as well as information about dual bound for problem or best solutions uploaded by other scientists. As a result, the solutions obtained with the examined algorithms can easily be compared to dual bound, or to other known solutions.

In UCFA experimentation system, a user can choose between MSA and MHU algorithms. Each of them has configurable set of parameters. In case of MSA, these parameters are: number of possible paths for demand – *KPath*, start temperature T_s, end temperature T_k, temperature interval τ, iterations number L. In case, of MHU the parameters are: *KPath*, and iterations number L.

Multiple tests can be run in parallel; the only limitation is performance of platform, on which the tests are being run. Each test can be paused and resumed or cancelled. At each stage of performing process the current results can be saved. Solution file format is XML, what facilitates easy integration with other simulators and benchmarks.

Results of tests are appended into summary file, where information on algorithm type, its parameters, solution cost, gap between dual bound and solution, simulation time, etc. are stored. Summary file can easily be used in multiple editors, due to its CSV structure.

VI. INVESTIGATION

Three complex experiments were conducted in order to tune algorithms' efficiency and carry out reliable comparison of MHU and MSA algorithms. These experiments concern:

- influence of *KPath* parameter in MSA algorithm,
- parameters setup in MSA algorithm,
- solution quality gain in MSA and MHU algorithms.

Each experiment was performed with the use of UCFA experimentation system. In the first and in the second experiment, the three network instances were used:

pdh, *dfn-bwin*, *nobel-german.*

Each test was repeated 5 times for a given set of parameters on a given network instance. In the third experiment, the six network instances were used:

dfn-bwin, *nobel-eu*, *nobel-germany*, *pdh*, *norway*, *dfn-gwin*.

In order to evaluate the algorithms, the two indices of performance were introduced:

- gap to dual bound,
- gap to base solution.

Both indices are expressed in percents. Equations (4) and equation (5) show how these measures are calculated.

$$\Delta_{BEST} = \left(\frac{F(current) - dualbound}{dualbound} \cdot 100\right)\%. \qquad (4)$$

$$\Delta_{FIRST} = \left(\frac{F(first) - F(current)}{F(current)} \cdot 100\right)\%. \qquad (5)$$

$F(current)$denotes the cost of a current solution, while $F(first)$ denotes the cost of the first solution, and *dualbound* is the cost of dual bound.

A. *KPath influence in MSA algorithm*

This experiment concerns influence of *KPath* parameter onto solution quality. This parameter determines number of different paths for each demand what is in direct correlation with number of possible routing combinations. Ten values of *KPath* values were considered. These values range from 1 to 10. Fig. 2 presents results of the performed experiment.

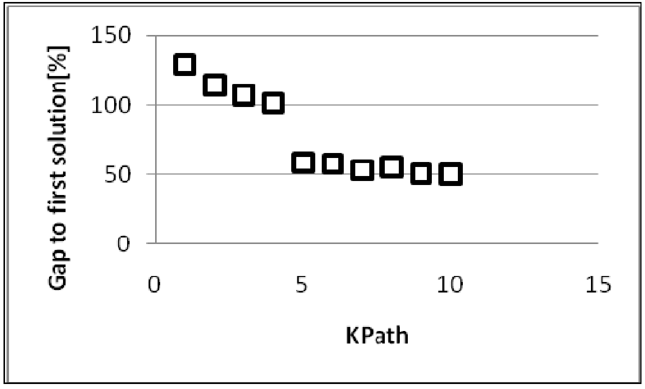

Figure 2. *KPath* influence in MSA algorithm.

One can observe that Δ_{FIRST} value is biggest for *KPath* equal to 1. However, this value must be rejected. It cannot be used, because in case when number of available paths for demand equals 1, then demand can flow along only one path, what means the flow is always same. When flow is always same, solved problem is no longer CFA problem, but only capacity assignment problem! Due to this fact, *KPath* cannot equal to 1.

Having rejected first best value, *KPath* parameters 2, 3 and 4 should be considered. All of them guarantee almost same solution efficiency gain (over 100%). *KPath = 4* seems to be the best choice, because it enhances quantity of possible demand flow combinations, while gap to first solution stays at same high level like when choosing parameters 2 or 3.

KPath values larger than 4 are not to be considered, because it can be easily noticed, that they vastly decrease solution quality. Furthermore, during experiment it was not always possible to find more than 4 different paths for each demand. *Nobel-germany* is an example of such network. If *KPath* parameter was set to value higher than 4, many networks would be unsolvable.

B. Parameters setup in MSA algorithm

This experiment concerns the parameters tuning for MSA algorithm. Eight parameter configurations were tested. Table 1 presents names of configurations mapped to parameters setup. It was assumed that each configuration should result in circa 100 000 total number of iterations. This assumption was made because of performance of available test platform. What is more it is assumed that temperature interval is always set to 0.9 and final temperature is of 1. The first assumption followed [10], but the second was done by authors of this paper on the basis of the results of preliminary experiments.

TABLE I. CONSIDERED MSA PARAMETERS CONFIGURATIONS.

Configuration	KPath	Ts	Tk	τ	L
Configuration 1	4	40000	1	0.9	1000
Configuration 2	4	4000	1	0.9	2000
Configuration 3	4	200	1	0.9	2000
Configuration 4	4	13	1	0.9	4000
Configuration 5	4	5	1	0.9	6000
Configuration 6	4	3.5	1	0.9	8000
Configuration 7	4	2.8	1	0.9	10000
Configuration 8	4	1.5	1	0.9	25000

The average gap to first solution Δ_{FIRST} was calculated for each configuration. Results are presented in Fig. 3. One can observe that configuration 8 delivers highest solution quality gain (133%). Configurations 5, 6 and 7 have very similar gain (over 130%). However, Configuration 1 gives only 104% of average improvement. This shows that slight difference in parameters setup results in big difference between qualities of solutions.

.

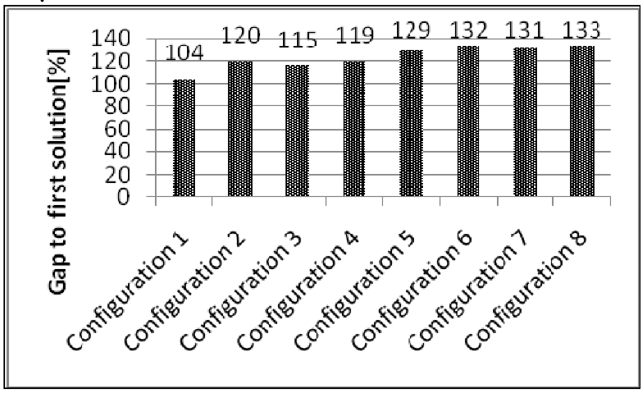

Figure 3. Average gap to first solution for MSA algorithm.

Before one of configurations is chosen as best one, one more calculation of measures (indices) is done. Optimistic cases (maximum gaps) for each configuration are shown in Fig. 4.

For all the configurations, maximum gaps are around 10% better than average gaps. The trend observed in previous chart remains same. Configuration 8 is best again; Configurations 5, 6, 7 have quite similar results to it. Although in this case difference between Configuration 8, and other configurations is higher than previously. Configuration 1 remains worst again.

Taking into consideration result presented above, Configuration 8 is chosen as the best one. Thus, all experiments concerning MSA algorithm are advised to be performed using Configuration 8.

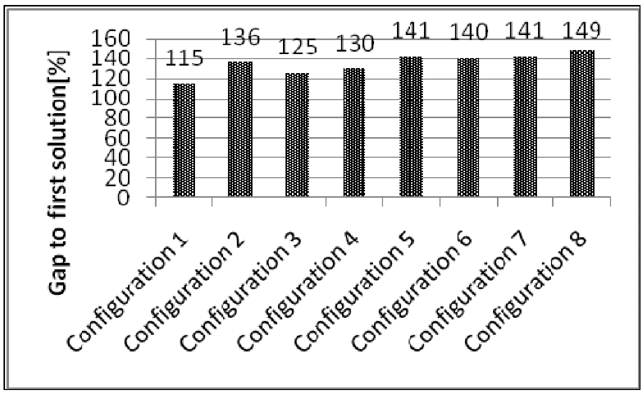

Figure 4. Maximum gap to first solution for MSA algorithm.

C. Solution quality gain in MSA and MHU algorithms

This experiment was designed in order to compare solution quality gain over first solution delivered by MSA and MHU algorithms. Parameters used for MSA are taken from previous experiment, e.g., configuration 8 was used. Both algorithms used *KPath* parameter set to 4.

Fig. 5 presents the results obtained with the use of MSA algorithm. Three cases are considered: optimistic, pessimistic and average. In the worst case, MSA guarantees solution quality gain of 121 %, what is not far from average gain of 138%. In the best case 163% gain is possible. It is worth to mention, that all these values are average values from all tested instances. Hence, it is possible, that in specific conditions these values can even be higher.

The same relation can be observed for results obtained with the use of MHU algorithm (see Fig. 6). Once more three cases are considered: optimistic (maximum), pessimistic (minimum) and the averaged. One can easily notice that solutions obtained with a use of MHU are of better quality than in the case of MSA algorithm. The pessimistic case is circa 50% better than optimistic in MSA algorithm. Average quality gain in MHU algorithm is 239%, but in best case it is possible to get 261% improvement of solution in comparison to the base solution.

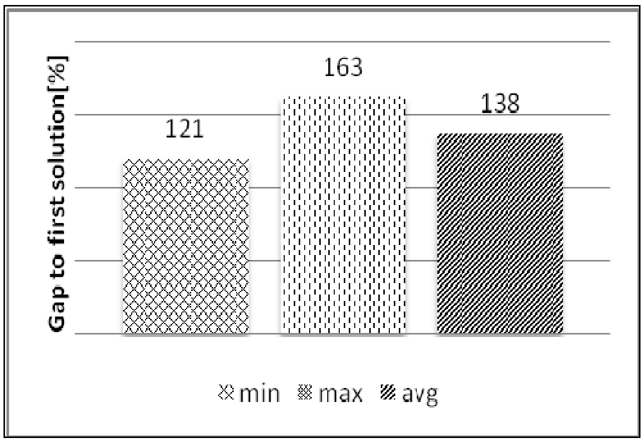

Figure 5. Minimum, maximum and average gap to first solution for MSA.

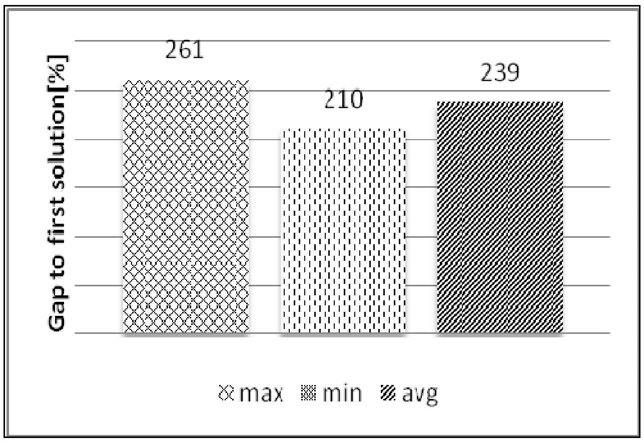

Figure 6. Minimum, maximum and average gap to first solution for MHU.

VII. CONCLUSION AND FUTURE WORKS

In this paper capacity and flow assignment problem, that is NP-complete [3], were discussed. For this purpose, two new algorithms have been proposed by the authors. Moreover, in order to conduct simulations, advanced experimentation system was designed and implemented.

Additional effort to tune parameters of created algorithms was taken. As a result, set of parameters that improve quality of solutions delivered by algorithms is proposed. Finally, both algorithms are examined to check solutions' quality gain over first solution. MSA delivers result which gave average gain of 138%, while MHU algorithm improves base solution 239% in average.

The results of both algorithms are satisfying. We believe that further solution improvement is still possible. Prospects for future are using ideas presented in [14] [15], including new configurations of parameters for MSA, or making simulation along with multistage experiment designs. It seems highly probable, that these two ideas will result in quality improvement of both algorithms.

REFERENCES

[1] A. Kasprzak, Designing of Wide Area Networks, Wroclaw University of Technology Press, (in Polish), 2001.

[2] B. Gendron, T. G. Crainic, and A. Frangioni, Multi-commodity Capacited Network Design, Telecommunications Network Planning, Kluwer, Norwell, MA, 1998, pp. 1-19.

[3] T. Cormen and R. Rivest, Introduction to Algorithms, Warsaw, 2004.

[4] J. Anisiewicz, T. Miksa, and M. Piec, "Cost optimization problem in Wide Area Network design," Proceedings of 10th Polish British Workshop, 2010.

[5] K. Walkowiak, "Ant Algorithm for Flow Assignment in Connection-Oriented Networks", Int. J. Appl. Math. Comput. Sci. 2, 2005, pp. 205-220.

[6] K. Walkowiak, "An Heuristic Algorithm for Non-bifurcated Congestion Problem", Paris, France, Proceedings of 17th IMACS World Congress, 2005..

[7] D. Corne, M. Oates and D. Smith, Telecommunications optimization: heuristic and adaptive techniques, John Wiley & Sons, 2000.

[8] Y. Azar and O. Regev, "Strongly Polynomial Algorithms for the Unsplittable Flow Problem", Proceedings of the 8th Conference on Integer Programming and Combinatorial Optimization, 2001, pp. 15-29.

[9] M. Gerla and L. Kleinrock, "On the Topological Design of Distributed Computer Networks", IEEE Trans Commun., Vol. COM-25, 1977, pp. 48-60..

[10] M. Pioro and D. Mehdi, Routing, Flow, and Capacity Design in Communication and Computer Networks, San Francisco, 2004.

[11] L. A. Wolsey, Integer Programming. Wiley-Interscience, New York, 1998.

[12] S. Orlowski, M. Pioro, and A. Tomaszewski, SNDlib 1.0--Survivable Network Design Library. [Online: 20 May 2011.]

[13] C. D. Gelatt, S. Kirkpatrick, and M. P. Vecchi M. P., "Optimization by Simulated Annealing", Science, New Series, Vol. 220, 1983, pp. 671-680.L. A. Wolsey, Integer Programming. Wiley-Interscience, New York, 1998.

[14] P. Bogalinski, I. Pozniak-Koszalka, L. Koszalka, and A. Kasprzak, "Computer System for Making Efficiency Analysis of Meta-heuristic Algorithms", Proc. of the 2nd ACIIDS conference, LNAI, vol. 5991, Springer, 2010, pp. 225-234.

[15] D. Ohia, L. Koszalka, and A. Kasprzak, "Evolutionary Algorithm for Congestion Problem in Computer Networks", KES 2009, LNAI, vol. 57, 2008, pp. 57-67.