Journal of Artificial Intelligence and Data Mining (JAIDM), x(x): xxx-xxx, xxxx

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

A Heuristic Algorithm for Multi-layer Network Optimization in Cloud Computing

Ali Hadian[1], Mehri Bagherian[2] and Behrouz Fathi Vajargah[3]

1. Department of Applied Mathematics, University campus 2, University of Guilan, Rasht, Iran,
2. Department of Applied Mathematics, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran,
3. Department of Statistics, Faculty of Mathematical Sciences, University of Guilan, Rasht, Iran,

Article Info

Article History:
Received 15 August 2020
Revised 02 February 2021
Accepted 25 March 2021

DOI:*10.22044/jadm.2021.9955.2133*

Keywords:
Model-driven Development, MPLS, Cloud Computing.

**Corresponding author: mbagherian@guilan.ac.ir (M. Bagherian).*

Abstract

One of the most important concepts in cloud computing is to model the problem as a multi-layer optimization problem, which leads to cost-savings in designing and operating the networks. The previous researchers have modeled the two-layer network-operating problem as an Integer Linear Programming (ILP) problem, and due to the computational complexity of solving it jointly, they have suggested a two-stage procedure in order to solve it by considering one layer at each stage. In this paper, considering the ILP model and using some of its properties, we propose a heuristic algorithm in order to solve the model jointly, considering the unicast, multicast, and anycast flows simultaneously. We first sort the demands in a decreasing order and use a greedy method in order to realize the demands in order. Due to the high computational complexity of the ILP model, the proposed heuristic algorithm is suitable for the networks with a large number of nodes. In this regard, various examples are solved by the CPLEX and MATLAB software. Our simulation results show that for the small values of M and N, CPLEX fails to find the optimal solution, while AGA finds a near optimal solution quickly. The proposed greedy algorithm could solve the large-scale networks approximately in polynomial time, and its approximation is reasonable.

1. Introduction

In the cloud computing area, the concept of the transfer layer or the data exchange layer could be modeled as a graph [1] . The two-layer network arises in the generic network architecture in the cloud computing area, i.e. considering the physical layer as the first layer and the network layer as the second layer. This concept is well-known as Multi-Protocol Label Switching (MPLS), which is frequently used by the network communications. The MPLS networks consist of two types of devices: Label Edge Router (LER), in which one can consider the set of LERs as E, and Label Switch Router (LSR), in which one can consider the set of LSRs as G. In MPLS, the packets transmit along a Label Switch Path (LSP) between LERs and LSRs [2]. Two models for MPLS-Traffic Engineering (MPLS-TE) have been presented in [3] and [4]. Reference [5] presents an MPLS technology that cloud improve the quality of an Information-Telecommunication System (ITS) network by creating the virtual channels between its nodes. In [6], an improved Particle Swarm Algorithm (PSO) for multi-objective-based optimization of the MPLS networks has been introduced. Due to the diversity of considerations in the physical layer such as the encryption and capsulation of data, the distance of the network points and the existence of different standards, different bandwidths can be generated using each one of these devices [7] and [8]. For example, according to the standards of the International Telecommunication Standardization

Unit (ITU-T), we can transmit a bandwidth of 10 Tbps to a distance of 1500 km (or more) using the optical waves [9], [10], [11], [12], and [13]. A real example of Cloud Computing is given in the [15]. In the computer networks, one of the famous problems is the flow allocation problem. Let D be the set of demands including all the unicast, anycast, and multicast demands in some network, which are stored in routers as the path tables including different types of flows as the known data of the problem [2]. We intend to allocate paths for these demands in the upper and lower layers of the network, minimizing the total allocation costs. The rest of the paper is organized as what follows. In Section 2, the problem is modeled as an ILP, and its complexity is discussed. The proposed heuristic algorithm is presented in Section 3. The numerical results and discussion are presented in Section 4, and Section 5 is devoted to the concluding remarks and the future research directions.

2. Problem Modeling

We considered the link set G for the first layer and the link set E for the second layer. Sending the flow per link in both layers incur costs, and the sum of these costs should be minimized, which forms the objective function of the optimization model. First, we defined the sets, constants, and variables similar to [2], as follows.

Sets:

- E Set of upper layer links;
- D Anycast, multicast, and unicast demands;
- D^{DS} Set of anycast downstream demands;
- $P(d)$ Set of all available paths for demand d;
- $Q(e)$ Set of all paths in G that link e on the upper layer uses them;
- G Links of the lower layer.

Constants:

- h_d Volume of demand d;
- ξ_e Routing cost unit on link $e \in E$;
- κ_g Routing cost unit on link $g \in G$;
- M Capacity module size of the upper layer link;
- N Capacity module size of the lower layer link;
- $\tau(d)$ Index of demand d. If d is a downstream demand, $\tau(d)$ must be an upstream demand, and vice versa.
- $s(p)$ Source node of path p.
- $t(p)$ Destination node of path p.

Variables:

$\delta_{edp} \in \{0,1\}$ = 1 if link e on path p is used to realize demand d; 0, otherwise.

$\gamma_{geq} \in \{0,1\}$ = 1 if link g on path q is used to realize link e on the upper layer; 0, otherwise.

$x_{dp} \in \{0,1\}$ = 1 if path p is used to realize demand d; 0, otherwise.

$y_e \in Z^+$ Capacity of the upper layer link e.

$z_{eq} \in Z^+$ Number of paths q that link e on the upper layer uses them.

$u_g \in Z^+$ Capacity of the lower layer link g.

Now the following ILP model could be formulated [2]:

$$Min\ F = \sum_{e \in E} \xi_e y_e + \sum_{g \in G} \kappa_g u_g \qquad (2-a)$$

S.t:

$$\sum_{d \in D} \sum_{p \in P(d)} \delta_{edp} x_{dp} h_d \le M y_e\ ,\ e \in E \qquad (2-b)$$

$$\sum_{e \in E} \sum_{q \in Q(e)} \gamma_{geq} z_{eq} h_d \le N u_g\ ,\ g \in G \qquad (2-c)$$

$$\sum_{p \in P(d)} x_{dp} s(p) = \sum_{p \in P(\tau(d))} x_{\tau(d)p} t(p), d \in D^{DS} \qquad (2-d)$$

$$\sum_{p \in P(d)} x_{dp} = 1\ ,\quad d \in D \qquad (2-e)$$

$$\sum_{q \in Q(e)} z_{eq} = y_e\ ,\quad e \in E \qquad (2-f)$$

The objective function (2-a) aims to minimize the cost of the capacity assigned in both network layers. Constraints (2-b)–(2-f) are the same as in the single layer network design problem. Equally, (2-e) ensures that each upper layer link is realized by a set of lower layer paths. Condition (2-f) states that the flow in each lower layer link cannot exceed its capacity.

Since the capacities of the modules in the upper and lower layers are restricted to M and N, respectively, the sum of the flows that could be allocated to the links in the layers should not exceed M (for the upper-layer) and N (for the lower-layer). The constraints (2-b) and (2-c) show this limitation. Equation (2-f) states that each link in the upper-layer can use multiple paths of the lower-layer, provided that the sum of the total streams for the upper-layer links used in the lower layer links does not exceed the total capacity of that link. Equation (2-d) states that each path can be used only as upstream or downstream. Equation (2-e) states that only one path must be allocated per flow. Due to the complexity of the multi-layer models (because of the existence of binary and integer variables), the heuristic algorithms are required in order to solve the larger problem instances. One approach is to tackle the optimization in all network layers jointly. However, since the routing and capacity decision variables of both layers are bound to each other,

this strategy may not be efficient. Another approach is to optimize each network layer in a separate phase, i.e. the problem in the upper layer is solved first, and the lower layer is optimized next. The algorithms developed for the single layer problems can be used, i.e. the methods proposed in [2]. If the algorithms used to optimize each layer are relatively fast, the procedure can be repeated for many times, and in each subsequent iteration, the information obtained in the previous iteration can be used to improve the performance of the overall optimization process.

This model was introduced in [2], and due to its complexities, it was suggested to optimize the problem in separate stages, i.e. the upper layer optimized first, and the lower layer optimized next. Since the two layers are dependent, this procedure should be down alternately to get a near optimal solution. If we denote the cardinality of each set D by $|D|$, model (1-a)-(1-f) has $2 \times (|E| + |D|) + |G|$ constraints and $|D| \times |P| + |Q| \times (|E| + 1) + |G|$ variables, making any exact algorithm of high computational complexity in solving large networks. In other words, the solvers like CPLEX or Gurobi fail to find the optimal solution for large-scale networks. Therefore, the heuristic procedures seem to be useful methods, which could find good solutions in a reasonable time for large scale networks. In this paper, we propose a heuristic algorithm based on a simple greedy strategy for solving Problem $(2-a)-(2-f)$ considering both layers jointly. We first sort the demands in a decreasing order and use a greedy method to realize the demands in order.

3. Heuristic Algorithm for Solving Model Jointly

As our investigations show, no algorithm has been proposed to solve model $(2-a)-(2-f)$ efficiently. Only in [2], it is suggested that each layer is solved separately with a greedy method or the Flow Deviation for Network Design (FDND) algorithm. Then using an iterative procedure, by optimizing each layer for several times, the information obtained in the previous iteration is used in order to improve the performance of the overall optimization process. We refer to this method as GRFA. Actually, GRFA considers the layers separately and ignores the two-layer dependency.

Note that the function Find_Best_Path d searches all the candidate paths available for demand d (line 4), and the function DC_Node $\tau(d)$ returns the DC node selected for demand $\tau(d)$ (line 6). Also note that the functions Find_Best_Path and Find_Best_Path_DC are generic, and can implement various strategies in order to find a routing path like the shortest path in a residual graph that only contains links with a residual capacity greater than the requested bit-rate of the considered demand.

Algorithm GRFA (Greedy algorithm for anycast, multicast, and unicast demand flow allocation) [2]

Input: set of edges E, set of anycast, multicast, and unicast demands D, sets $P(d)$ including candidate paths for each demand $d \in D$.

Ensure: path selection (routing) for each demand $d \in D$ included in set X, value of objective function.

1: **procedure** GRFA(D, P(d))
2: **for** d ∈ D **do**
3: **if** *Is_Not_Allocated(τ (d))*, **then**
4: *p := Find_Best_Path (d)*
5: **else**
6: *p := Find_Best_Path_DC(d,DC_Node (τ (d)))*
7: **end if**
8: $X \coloneqq X \cup x_{dp}$
9: *D := D - {d}*
10: **end for**
11: **end procedure**

Now we propose a heuristic algorithm to solve Problem $(2-a)-(2-f)$ using a Greedy based strategy. Our proposed algorithm, which we call it the Adaptive Greedy Algorithm (AGA), solves the model considering both layers simultaneously.

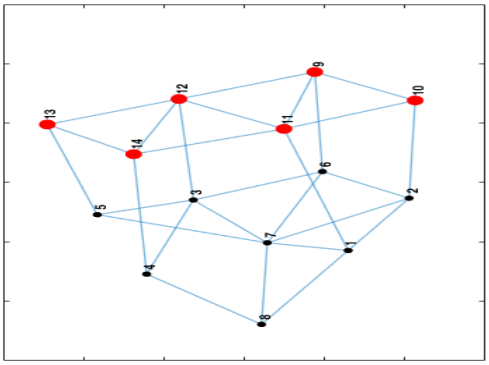

Figure1. An example of the MPLS network.

Note that as some advantages of the greedy algorithm, we can refer to its adaptability to the means of ordering and routing strategies, its relatively low complexity, and short execution time. Before explaining the algorithm, some important remarks on model $(2-a)-(2-f)$ are noticed. Each upper-layer node is connected to one lower-layer node by a physical link. Therefore, there is no demand d such that its source or destination nodes are on the lower-layer. On the other hand, one or more paths in the lower-layer connect every two nodes on the upper-layer. For example, in Figure 1, if one wants to send

data from node 13 to node 14 on the upper-layer, the data must be sent on one of the many available paths from node 5 to node 4 on the lower-layer.
As it can be seen in Figure 1, there are 22 paths from node 5 to node 4. The lengths of these paths vary from 3 to 8. Table 1 shows all of these paths for this example from node 13 to node 14. We denote this collection of paths by $Q(e)$, where e is the link between nodes 13 and 14.

Table 1. All paths from node 13 to 14.

13	5	3	4	14					
13	5	7	3	4	14				
13	5	7	8	4	14				
13	5	7	1	8	4	14			
13	5	7	6	3	4	14			
13	5	3	7	8	4	14			
13	5	7	2	1	8	4	14		
13	5	3	7	1	8	4	14		
13	5	7	2	6	3	4	14		
13	5	3	6	7	8	4	14		
13	5	7	1	2	6	3	4	14	
13	5	3	6	2	1	8	4	14	
13	5	3	7	2	1	8	4	14	
13	5	7	6	2	1	8	4	14	
13	5	3	6	7	1	8	4	14	
13	5	3	6	2	7	8	4	14	
13	5	7	8	1	2	6	3	4	14
13	5	7	3	6	2	1	8	4	14
13	5	3	7	6	2	1	8	4	14
13	5	3	6	7	2	1	8	4	14
13	5	3	6	2	7	1	8	4	14
13	5	3	6	2	1	7	8	4	14

Since every link e in the upper-layer has several paths on the lower-layer, and each demand is produced and transferred on the upper-layer, for each demand d, we have several paths on the lower-layer from which we search for the shortest one to realize demand d. We use this observation to propose our heuristic greedy algorithm, AGA.

In order to get the complexity of the algorithm, it is enough to obtain the number of iterations in the loops of the algorithm. Line 2 sorts the elements of D, which is done in $O(|D|\log(|D|)$. Lines 4 to 19 are repeated $|D|$ times. It is also possible to find the best route with a linear search, so lines 4 to 8 are repeated for a maximum of $|P|$ and line 10, $|Q|$ times. In order to find the upper-layer paths in the lower-layer, all the edges of the path in the upper-layer must be compared with all the edges of the lower layer paths; this can be done by the $|E| \times |G|$ iterations. Lines 12 and 16 can also be done by p and q iterations, respectively. This discussion leads us to the complexity of the AGA as $(|D|^2 + |D| \times (2 \times |P| + |E| \times |G| + 2 \times |Q|))$, which is obviously polynomial time.

In order to solve by AGA, we first sort demands D by the value of each demand descending. This will meet the demand for larger values sooner. In order to meet any demand, we first find the best of all possible supply paths at the upper layer. Since each arc in the upper layer uses a path from the lower layer, we find all the possible paths in the lower layer that the selected path from the upper layer uses, and keep in set Q. Then we choose the best path in Q. According to the greedy algorithm, the chosen path is the best path by which demand d can be met. Therefore, we calculate the values of z and the amount of flow that must be transferred in this path according to the volume of this demand and add the selected path to the solution set x.

Algorithm: Adaptive greedy algorithm (AGA) to solve model $(\mathbf{1}-\boldsymbol{a})-(\mathbf{1}-\boldsymbol{f})$

Input: set of edges E and G, set of anycast, multicast and unicast demands D, sets $P(d)$ including candidate paths for each demand $d \in D$.
Ensure: path selection (routing) for each demand $d \in D$ included in set X, value of objective function.

1: **procedure** *AGA* $(D, P(d))$
2: *sort descending* (D)
3: **for** $\boldsymbol{d} \in \boldsymbol{D}$, **do**
4: **if** *Is Not Allocated*$(\tau(d))$, **then**:
5: p := *Best Path On E for demand d*
6: **else**:
7: p := *Best Path On E for demand d On DC node from* $\tau(d)$
8: **end if**:
9: $X := X \cup \{x_{dp}\}$
10: $Q(p)$:=*Compute all paths on G for all e on p*
11: q=*find best path on* $Q(p)$
12: **for** $e \in p$ **do**
13: z_{eq}:=*cardinal of* $Q(e)$
14: *compute* $y(e)$ *according to* $h(d)$
15: **end for**
16: **for** $g \in q$ **do**
17: *compute* $u(g)$ *according to* $h(d)$
18: **end for**
19: $D := D - \{d\}$
20: **end for**
21: **end procedure**

In order to show the performance of AGA, we compare the algorithm results with the exact solutions for a few small examples solved by the CPLEX optimization software. Due to the complexity of the model, CPLEX could solve only small problem instances to optimality, so in the next section, several small problem instances are simulated and solved by AGA as an approximation algorithm and by the CPLEX optimization software as an exact solver, and the results obtained are compared and discussed. By this comparison, we show that the solutions obtained by AGA are not far from the exact solutions.

4. Numerical Results and Discussion

In this section, we create several small MPLS network instances with known demands by simulation, and solve them by CPLEX. Then we solve these examples by AGA, and compare the solutions obtained. These examples are solved using a computer with an Intel Core i7-6700HQ processor and 12GB Ram. The simulated demands, MPLS graphs, and sets of paths are created and computed by MATLAB. AGA is implemented in the MATLAB software as well.

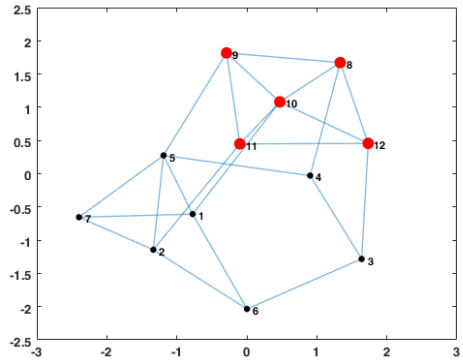

Figure 2. An example of an MPLS network.

First, we explain the example of Figure 2. In this MPLS network, we have 5 nodes on the upper-layer and 7 nodes on the lower-layer. Note that if M is a small number, Equation (1-b) is satisfied for a large amount of y_e. In this example, we consider the set of demands as Table 2. Therefore, if we set $M = 1$ and $N = 1$, the value of 35510 is obtained for the objective function. On the other hand, if we set M =30 and $N = 40$, then the value of the objective function becomes 297.

We solve this example by both CPLEX and AGA for different values of M and N, and show the results in Figure 3. The values of M and N are $M = 1\ to\ 8$ and $N = 1\ to\ 50$. In Figure 3, the vertical axis represents the value of the objective function, and the two horizontal axes represent the values of M and N. CPLEX finds the exact solution, and AGA obtains the approximate solution. Of course, the solutions do not exactly match but they are not far apart. For example, when $M = 3$ and $N = 43$, the exact optimal objective function is 1388, and AGA finds 1392 as a near optimal objective function.

Table 2. Simulated demands on MPLS of Figure 2.

Source	10	9	10	12	9	10	11	10	11	10	9	9
Destination	11	10	8	8	10	11	9	11	9	12	10	10
Value of h_d	4	4	9	6	7	8	5	3	6	4	9	9

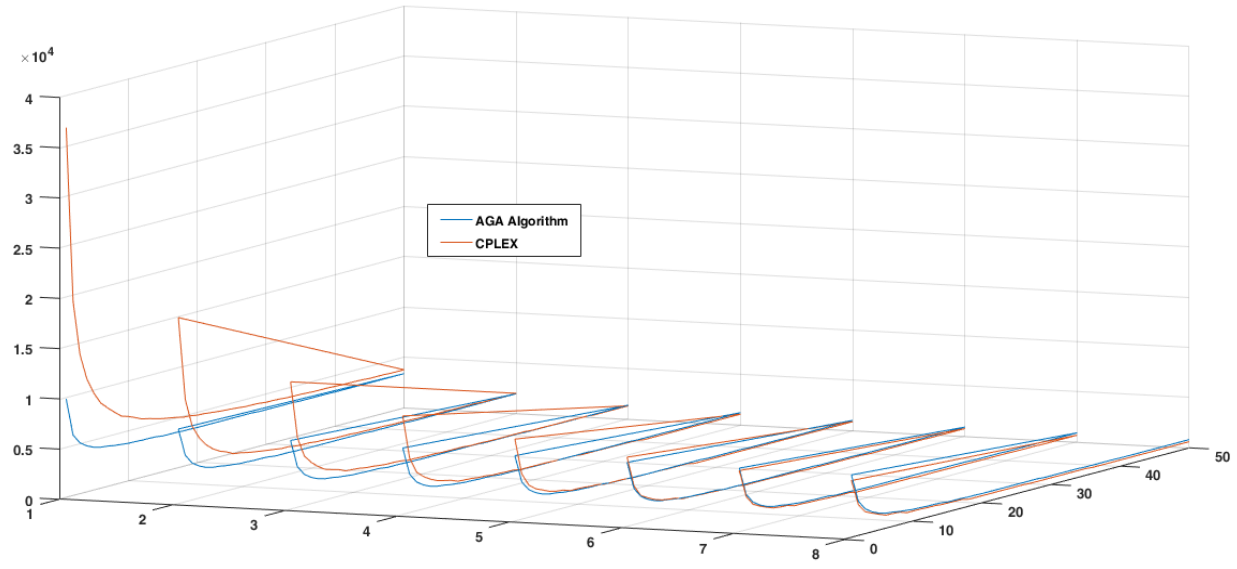

Figure 3. Comparing CPLEX with AGA.

Table 3 shows nine MPLSs with different sizes from small to large. Also Table 3 shows the time to get the answer for several values of M and N. CPLEX cannot solve examples 6 to 9 for some values of M and N or it takes a long time.

As shown in Table 3, CPLEX fails to get a solution for large size MPLS networks or it requires a very long time to get the optimal solution, while AGA finds a near optimal solution quickly. This fact is shown in Figure 4.

Table 3. Numerical results for 9 examples.

Example	1	2	3	4	5	6	7	8	9
Number of upper layer edges	3	8	11	12	14	11	15	18	17
Number of upper layer nodes	3	5	7	8	9	8	8	10	9
Number of lower layer edges	5	10	12	12	14	15	17	17	17
Number of lower layer nodes	4	7	8	8	9	9	10	10	10
Number of demands and maximum number of paths between 2 nodes	6	12	18	18	34	58	72	129	84
Number of all paths on upper layer	12	156	472	662	1146	392	2326	6794	5626
Number of all paths on lower layer	38	350	674	662	1104	2208	4260	4250	4522
Cardinality of Q_e	10	67	115	147	192	349	739	909	919
Average time to get the approximate solution by AGA implemented for different M and N in MATLAB (s)	0.16	0.4	0.32	0.55	0.5	1.39	1.76	2.881	1.24
Average time to get the exact solution for different M and N by CPLEX (s)	0.09	0.5	0.99	2.91	27	8136 Or fail	33335 Orfail	66687 Or fail	75160 Or fail

5. Conclusions and Future Research Works

In this paper, we presented a greedy algorithm to solve the multi-layer network model, which joins joined the two layers. Since the integer-programming model for the large-scale networks had a high computational complexity, the optimization softwares like CPLEX fail to solve the large-scale networks, while the proposed greedy algorithm could find a near optimal solution in polynomial time. On the other hand, for small values of M and N, CPLEX fail to find the optimal solution, while AGA find a near optimal solution quickly. In the future, we intend to solve the model with other heuristic algorithms such as the Particle Swarm Optimization (PSO) method. We also intend to compare the AGA and PSO algorithms and discus their specific properties.

Financial support: This research work did not receive any specific grant from the funding agencies in the public, commercial or not-for-profit sectors.

Conflict of interest: The authors declare that they have no conflict of interest.

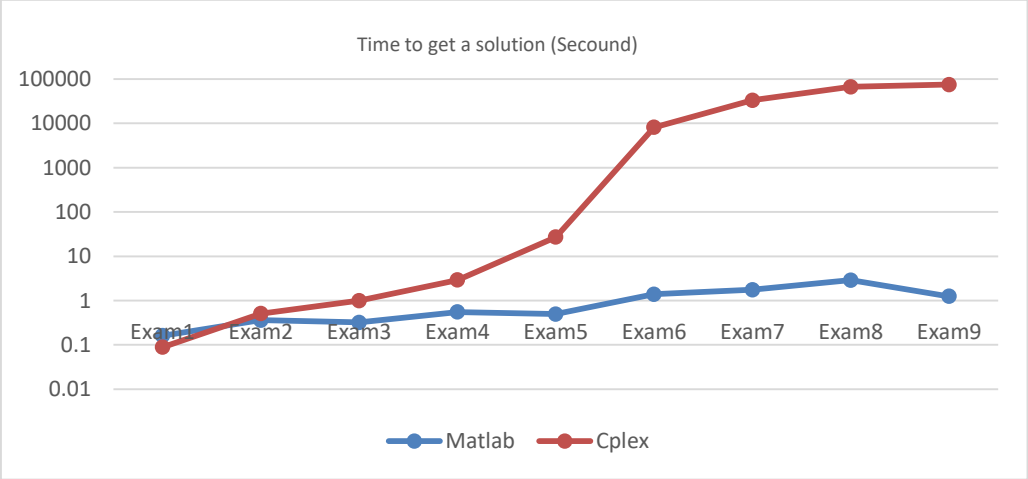

Figure 4. Comparing the run times of the two methods.

References

[1] M. Pióro, and D. Medhi, *Routing, flow, and capacity design in communication and computer networks*. Elsevier, 2004.

[2] K. Walkowiak, *Modeling and optimization of cloud-ready and content-oriented networks*, Springer International Publishing, vol 56, 2016.

[3] G. Carofiglio, G. Morabito, L. Muscariello, I. Solis, and M. Varvello, *From content delivery today to information centric networking*. Computer Networks, vol *57*(16), pp. 3116-3127, 2013.

[4] G. Tyson, E. Bodanese, J. Bigham, and A. Mauthe, *Beyond content delivery: Can icns help emergency scenarios?* IEEE Network, 28(3), pp. 44-49, 2014.

[5] J.M. Simmons, *Optical network design and planning*. Springer, 2014.

[6] I. Tomkos, S. Azodolmolky, J. Sole-Pareta, D. Careglio, and E. Palkopoulou, *A tutorial on the flexible optical networking paradigm: State of the art, trends, and research challenges*. Proceedings of the IEEE, 102(9), pp. 1317-1337, 2014.

[7] Y. Li, D. King, F. Zhang, and A. Farrel, *Generalized Labels for the Flexi-Grid in Lambda Switch Capable (LSC) Label Switching Routers*. IEEE, 2015.

[8] B. Mukherjee, *WDM optical communication networks: progress and challenges.* IEEE Journal on Selected Areas in communications, 18(10), pp. 1810-1824, 2000.

[9] Recommendation, I.T.U.T. *Optical interfaces for multichannel systems with optical amplifiers*. *Vol. G*, *692*, 1998.

[10] O. Nevzorova, O. Lemeshko, A. Mersni, A.M. Hailan, A.S. Ali, and S. Harkusha, *July. Improved Two-Level Method of Multicast Routing in MPLS-TE Network*. IEEE 2nd Ukraine Conference on Electrical and Computer Engineering (UKRCON) (pp. 846-850). IEEE, 2019.

[11] I. Zhukovyts'kyy, V. Pakhomova, H. Domanskay, and A. Nechaiev, *Distribution of information flows in the advanced network of MPLS of railway transport by means of a neural model.* In MATEC Web of Conferences (Vol. 294, p. 04007). EDP Sciences. 2019.

[12] M. Masood, M.M. Fouad, R. Kamal, I. Glesk, and I.U. Khan, *An improved particle swarm algorithm for multi-objective-based optimization in MPLS/GMPLS networks*. IEEE Access, 7, pp. 137147-137162, 2019.

[13] M. Masood, M. Mosta Foyad, R. Kamak, I. Glesk, and I. Ullahkhan, *An Improved Particle Swarm Algorithm for Multi-Objective-based Optimization in MPLS/GMPLS Networks*, IEEE Access, 2019.

[14] A.R. Sharafat, S. Das, G. Parulkar, and N. McKeown, *Mpls-te and mpls vpns with open-flow.* In Proceedings of the ACM SIGCOMM, pp. 452-453, August 2011.

[15] MABHOOT, Nahid; MOMENI, Hossein. *An Energy-aware Real-time Task Scheduling Approach in a Cloud Computing Environment*. Journal of AI and Data Mining, 2021.