Experimental Evaluation of Algorithms for Packet Routing in Software Defined Network ⋆

Pavel Borisovsky * **Anton Eremeev** * **Sergei Hrushev** *
Vadim Teplyakov **

* *Sobolev Institute of Mathematics SB RAS, Novosibirsk, Russia (e-mail: eremeev@ofim.oscsbras.ru).*
** *Yaliny Research and Development Center, Moscow, Russia*

Abstract: In this paper, we consider a manufacturing flow-line organized as a series-parallel system of machines separated by finite buffers. The failure and repair times of machines are supposed to be exponentially distributed. The production rate of each machine is deterministic, and different machines may have different production rates. The buffer allocation problem consists in determining the buffer capacities with respect to a given optimality criterion, which depends on the average production rate of the line, the buffer acquisition and installation cost and the inventory cost. The tentative solutions are evaluated with an approximate method based on the Markov models aggregation approach. The computational experiments show better quality of solutions obtained by a genetic algorithm compared with the local descent and tabu-search algorithms. It is indicated that in many test problems several clusters of local optima can be found.

Keywords: routing problem, software defined network, greedy algorithm, FPTAS, computational experiment.

1. INTRODUCTION

When designing production systems such as automatic lines, flexible production systems or automated assembly lines in which parts are moved from one machine to another another with the help of some transport mechanism, the following problem arises.

Software Defined Networks attempt to centralize communication network intelligence in one network node (Control Center) by separating the forwarding process of data packets from the routing process, which is performed in the Control Center. In this paper, we aim at packets routing in prospective Software Defined Satellite Networks (SDSN) (see e.g. Xu et al. (2018); Tang et al. (2014)) that provide world-wide telecommunication services. Such network consists of satellites and ground stations (or base stations) which provide the gates to Internet and communication with satellites. One of the ground stations is the Network Operations Control Center (NOCC). The packet routes for each source-destination pair (s_i, t_i), are computed at NOCC in real time and each node (satellite or ground station) regularly receives the updated routs for all packets that originate in this node. Each packet sent from s_i to t_i contains some content data and a path of the packet route from s_i to t_i. An upper bound L on the number of edges in packet paths is imposed due to a technical limitation on the number of bits reserved for encoding a packet route. Short packet paths also tend to have low delay. For simplicity we assume that each problem instance describes the system in a single time-frame and all demands for the time-frame are known in advance.

In this paper, the packet routing problem in software defined network is formulated (in what follows, we call it "packet routing problem" for short) and a greedy algorithm is proposed for its approximate solution. An alternative approach to approximate solving this problem by means of reduction to a special case of fractional length-bounded maximum multicommodity flow is suggested. Experimental comparison of the greedy algorithm and a fully polynomial-time approximation scheme (FPTAS) for fractional length-bounded maximum multicommodity flow is carried out on the testing instances, representing prospective SDSN.

1.1 The Packet Routing Problem Formulation

The packet routing problem has the follwoing input data:

- $G = (V, E)$ is a digraph of inter-satellite and satellite-ground connections, $|V| = n$, $|E| = m$, where E is the set of arcs, V is the set of all satellites and ground stations in the system.
- $u(e) \geq 0$ are the bandwidths of links (arcs).
- N is the total number of user-to-user sessions.
- The sessions are represented as triples (A_i, B_i, W_i), $i = 1, \ldots, N$ where $A_i, B_i \in V$ is the source-destionation pair of nodes, W_i is the data traffic per time unit along this session. A real-life user-to-user session is considered here as session between the nodes (sattelites or a ground stations) to which the pair of users is currently connected.

⋆ The research is supported by Russian Science Foundation grant 21-41-09017.

- $L_{\max}$ is an upper bound on the number of edges in packet paths
- $\tau(e)$ is the integer-valued delay while transmitting data along the link $e \in E$.

It is required to find routes in graph G for the maximum possible number of sessions, so that each session is mapped to no more than one route, taking into account the restriction on the maximum number of arcs in the path and that for each arc $e \in E$ the total amount of transfered information on all routes passing through e should not exceed its bandwidth $u(e)$. As a secondary criterion, the maximum delay among all routed sessions may be considered.

The problem formulation with an upper bound on the secondary criterion would be NP-hard, which follows from the NP hardness of the fractional length-bounded maximum multicommodity flow problem Baier (2003), see more details in Section 3 below. In practice, the problem has to be solved fast in real time, this is why the packet routing problem is defined with maximization of the main criterion only. The secondary criterion in what follows will be taken into account optionally, ony when developing heuristics.

1.2 Contribution of the Paper

Our main contribution consists in formulation of the packet routing problem as a fractional length-bounded maximum multicommodity flow problem where all edges have the unit length. Investigation of the practical performance of a simple greedy heuristic and of the FPTAS from Borisovsky et al. (2019) with respect to the secondary criterion and the required CPU time is another contribution of the paper. Therefore it may be considered as a more detailed presentation and study of the practical instances considered in Borisovsky et al. (2019).

2. A GREEDY ALGORITHM FOR PACKET ROUTING PROBLEM

The main idea of the greedy algorithm is to sequentially assign sessions to the shortest delay routes between the sender node and the receiver node. Once a new session (A_i, B_i, W_i) is assigned, the throughputs of edges on a shortest path from A_i to B_i are decreased by W_i. When the capacity of an edge is exhausted, this edge is removed from further consideration. To find the shortest paths between all pairs of vertices, one can use one of the well-known algorithms, e.g. the Dijkstra's, the Floyd-Warshall (see, e.g., Cormen et al. (2001)) or the Bellman-Ford algorithm. Since the set of shortest paths is computed at most m times, the time complexity of the greedy algorithm is $O(m^2 \log n)$ if the Dijkstra algorithm with heaps is used. The Bellman-Ford algorithm may be easily truncated so that the returned set of paths consists has at most $L_{\max}$ edges in each path. In such a case the time complexity of the greedy algorithm is $O(m^2 L_{\max})$.

The sequence of sessions assignment is in the descending order of delay along the shortest path from A_i to B_i. This rule is chosen in order to reduce the maximum delay and fulfill the restriction on the maximum number of arcs in path $L_{\max}$, because sessions with the greatest delay on the shortest paths would most likely violate the $L_{\max}$ constraint. It is these sessions that are routed first when there is a relatively large margin of arc capacity.

The greedy algorithm does not have a guarantee of accuracy and the resulting set of routes may violate the limit on the maximum number of arcs in a path $L_{\max}$, unless the Bellman-Ford algorithm is used in it. However, as the computational experiment in section 5 will show, in practice it has competitive results in terms of the quality of solutions and has a small running time.

3. RELAXED PACKET ROUTING PROBLEM

This section contains the statement of the relaxed routing problem, which is simplified version of the original problem, as well as a description of the approximate algorithm for solving it with any a-priori given accuracy. The relaxation of the original packet routing problem consists in skipping the requirement that the whole traffic W_i of each session i has to be routed via a single path. This implies that instead of considering specific sessions we can group all traffic between each source and destionation nodes into one *demand* and distribute it optionally between a certain number of paths, connectiong the two nodes.

Problem input data:

- $G = (V, E)$ is a digraph, $|V| = n,\ |E| = m$,
- $u(e) \geq 0$ are the bandwidths of arcs,
- k is the number of source-destionation pairs of nodes $(s_j, t_j) \in V^2$, $1 \leq j \leq k$.
- d_j is the amount of data per unit of time, requested for transmission from the node s_j to the node t_j;
- $L_{\max}$ is an upper bound on the number of edges in packet paths.
- $\tau(e)$ is the integer-valued delay on the link $e \in E$.

A feasible solution is a set of paths for all k source-destionation pairs with indication of real-valued transmission volumes for each path. Costraints:

- The sum of all transmission volumes along the paths passing through each arc $e \in E$ must not exceed the capacity of the arc.
- In sum, the amount of transmission over paths starting at s_j and ending in t_j does not exceed d_j.

The optimization criterion is to maximize the total amount of data transfer across all source-destionation pairs. Let us denote:

- P is a directed path in graph G,
- S is a set of all nodes with positive amount of data, requested for transmission from that node, i.e. $S = \{v \in V\ :\ \exists\ j,\ s_j = v\}$,
- $T_v \subset V$ is the set of all nodes with positive amount of data, requested for transmission from v to that node, i.e. $T_v = \{v'\ :\ \exists\ j,\ s_j = v,\ t_j = v'\}$.

Let $\mathcal{P}_j(L)$ be the set of paths with at most L arcs, connecting the nodes s_j and t_j. In what follows, $\mathcal{P}(L) = \cup_j \mathcal{P}_j(L)$. The variable $x(P)$ will be the amount of data transfer per time unit along the path P. Now the relaxed routing problem may be formulated as follows:

$$\max \sum_{P \in \mathcal{P}(L_{\max})} x(P), \tag{1}$$

$$\sum_{P\in\mathcal{P}(L_{\max}):e\in P} x(P)\le u(e),\quad e\in E, \tag{2}$$

$$\sum_{P\in\mathcal{P}_j(L_{\max})} x(P)\le d_j,\quad j=1,\dots,k, \tag{3}$$

$$x(P)\ge 0,\quad P\in\mathcal{P}(L_{\max}). \tag{4}$$

This linear programming problem (let us denote it by **P**), generally speaking, contains an exponential number of variables, however, it can be used to build algorithms without a need to process and store all of the variables. This problem is known as *fractional length-bounded maximum multicommodity flow problem* where all edges have the length equal to one, see Baier (2003); Borisovsky et al. (2019). Note that the fractional length-bounded maximum multicommodity flow problem with arbitrary edge lengths is NP hard (see Baier (2003)), which implies that imposing a constraint on maximal communication delay would make the relaxed problem intractable.

A modification of this problem with an additional constraint that the flow on all edges must be integer-valued is called integral length-bounded maximum multicommodity flow. The results from Garg et al. (1997) imply that even when there is no length constraint at all, this problem does not admit approximation algorithms with constant approximation ratio, unless P=NP. This implies intractability of the (non-relaxed) packet routing problem, even in the special case where all sessions i require identical data traffic W_i.

If the number of sessions in each source-destination pair (s_j, t_j) is large, the routing problem may be solved approximately, using a feasible solution to the relaxed problem **P**. To do this, in case all sessions have equal transfer rates it suffices to solve (exactly or approximately) the relaxed version of the problem, where d_j is equal to the total transfer rate of sessions from vertex s_j to t_j and to round down the obtained solution w.r.t. the transfer rate of a single session. If sessions have different rates, then this issue can be resolved analogously.

3.1 Fully Polynomial Time Approximation Scheme

Due to the large number of variables of the problem **P**, instead of finding its exact solution, one can consider the problem of finding approximate solution that differs in objective function from the optimum β of the **P** problem at most by a factor $1-\omega$, where $\omega\in(0,1)$ is a specified parameter for the required precision.

A $(1-\omega)$-approximation algorithm is an algorithm for solving a maximization problem, that obtains a feasible solution with the value of the objective function f_{appr} that differs from the optimal f^* by no more than $(1-\omega)$ times if the problem is solvable:

$$f_{appr}\ge(1-\omega)f^*.$$

Fully polynomial time approximation scheme (FPTAS) for the maximization problem is a family of $(1-\omega)$-approximation algorithms for all $\omega>0$ with polynomially bounded running time with respect to the length of the problem input $|x|$ and w.r.t. $1/\omega$ (see, e.g. Garey and Johnson (1979)).

An FPTAS for the relaxed routing problem is based on the same principles as FPTAS of Fleischer (2000) for the problem of maximum multi-product flow. Here approximate solution to the problem **P** is found via iterative refinement of the existing primal solution and at the same time an approximate solution of the dual problem is calculated and updated. The latter, due to the duality inequality, allows to estimate the error of the available primal solution on each iteration of the algorithm.

Instead of the current dual-feasible solution in the algorithm it is more convenient to calculate a set of parameters $\{\ell(e)\}_{e\in E}$, called *arc lengths*, related to the variables of the dual problem by multiplication with some scaling factor α. The number of times the factor α is updated is $r_{max}:=\lfloor\log_{1+\varepsilon}\frac{1+\varepsilon}{\delta}\rfloor$. The value of parameters $\varepsilon,\delta>0$ will be defined later. In what follows, $\ell(P)$ will denote the sum of the lengths of all arcs that make up the path P.

The algorithm starts with length function $\ell(e)=\delta$ for all $e\in E$, and with a primal solution $x(P)=0$. While there is a path of length less than 1, the algorithm selects such a path and updates the primal and the dual variables as follows. For the primal solution x, the algorithm increases the flow along path P by the minimum edge capacity in the path, scaled down by the factor $r_{\max}$. Let us denote this bottleneck capacity by u. Now the dual variables are updated in such a way that the higher the congestion of an edge the greater multiplier is given to its length:

$$\ell(e)=\ell(e)\left(1+\frac{\varepsilon u}{u(e)}\right),\ e\in P.$$

Note that the problem **P** for $L_{\max}=\infty$ may be reduced to the maximum multicommodity flow problem in a new graph G', where for each vertex $v\in V$ for $|T_v|>0$, we indroduce $|T_v|$ dummy vertices, each of which is incident to exactly one arc leading from this dummy vertex to v. The dummy vertices one-to-one correspond to the recipient vertices $t\in T_v$ and the capacities of the arcs, connecting them to the vertex v, are equal to d_j, where j is such that $s_j=v$, $t_j=t$.

With this reducibility taken into account, the problem **P'** may be solved with any given accuracy $\omega>0$ by modifying the algorithm from Fleischer (2000). The main differences from the original algorithm are the following:

- The subset $\mathcal{P}(L_{\max})$ is used instead of a set of all paths from senders to recipients.
- The search for the shortest paths in $\mathcal{P}_i(L_{\max})$ is done by means of a truncated version of the Ford-Bellman algorithm.
- We choose $\varepsilon=\frac{3-\omega-\sqrt{(3-\omega)^2-4\omega}}{2}$, $\delta:=\frac{1+\varepsilon}{\sqrt[\varepsilon]{(1+\varepsilon)L_{\max}}}$.
- Graph G' is not explicitly used. For the storage of weights of arcs outgoing from dummy vertices, additional variables $\ell_j,\ j=1,\dots,k$ are introduced.

Justification of the approximation accuracy $1-\omega$ for the FPTAS, as well as admissibility of the resulting solution is provided in Borisovsky et al. (2019).

The sessions from each source node s_j to a target node t_j are routed within the amount $x(P)$, which are found in solving the relaxed problem. One can use the greedy algorithm as an attempt to improve the obtained solution

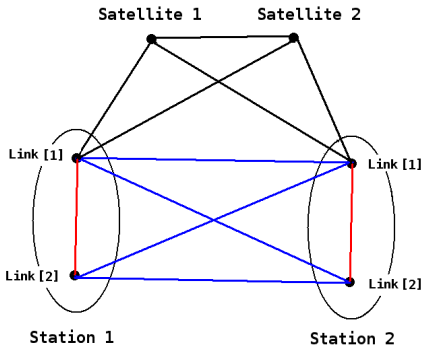

Fig. 1. Graph "Clique for links"

by routing those sessions that could not be routed within the amounts $x(P)$.

4. COMPUTATIONAL EXPERIMENT

The input data of the SDSN instances may be generated with different levels of details, regarding the communications of the ground stations. The considered options are described in the following subsections.

4.1 Graph "Clique for links".

The input data is presented as a list of satellites and stations, and connections between them. The satellite corresponds to one vertex, while the station consists of several links, and therefore it is represented by several nodes. The "satellite-satellite" and "satellite-station" arcs are given, specifying bandwidth and communication delay.

The "station-satellite" arcs are defined in the same way. The connections between stations are set for individual links, i.e. arcs are defined as pairs of links. For each such arc, only the delay is defined, and the total bandwidth is set for the entire link (one can distinguish incoming and outgoing bandwidth, but at the moment they are considered the same).

Thus, the graph consists of two types of vertices: satellites and links. The "satellite-to-satellite" arcs are assigned the bandwidth and delay. Connections between satellites and stations can be represented by arcs between the satellite and one of the links of the station, such arcs are also given the bandwidth and delay. All possible arcs of the "link-link" type, with the value of delay, and the total bandwidth may be specified for each link. It is also assumed that the links of one station are connected in a chain by arcs with bandwidth ∞ and delay 0. An examplary fragment of such a graph is shown in Fig. 1.

4.2 Graph "Large clique for links"

In the packet routing problem, it is assumed that capacities are given only for arcs, not for vertices. To build a graph with such property, it is necessary to add two more dummy vertices to each link. One of these vertices is responsible for incoming and the other one is responsible for the outgoing connections. These arcs will be assigned link bandwidths, and the delay value equal to 0.

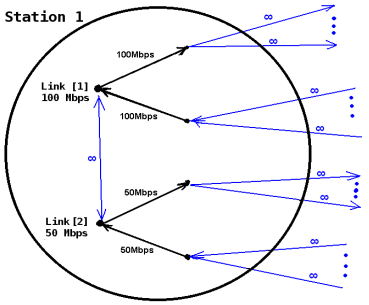

Fig. 2. Graph "Large clique for links' that models link bandwidths as arc capacities

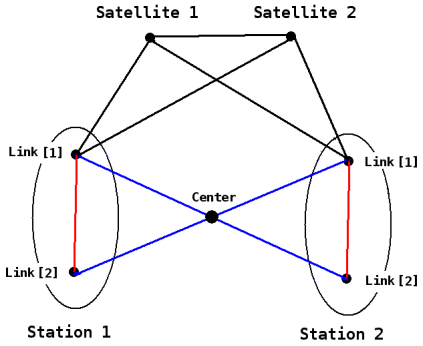

Fig. 3. Graph "Star' for individual links"

In what follows, the graph constructed in such a way will be referred to as a "link clique graph" since all dummy vertices must be interconnected (and hence form a clique). An examplary fragment of such graph is given in Fig. 2

Despite the fact that the described approach allows the most accurate description of the communication structure, due to high dimension of the input data, it seems to be irrelevant for practical applications.

4.3 The graph "Star for individual links"

To reduce the dimension of the problem and speed up the algorithms it is proposed to consider a simplified model where a vertex "Center" is added and connected to all links. Bandwidths of these arcs are equal to the bandwidth of the links, and the delay is calculated as the average delay from the current link to the rest. It is obvious that this way some information about individual delay values is lost. However, the solution built on such a graph, may be a good approximation of the optimum.

4.4 Graph"Star for Stations"

A further simplification of the problem is to ignore separate links, but combine them into one vertex corresponding to one station. As before, the "Center" vertex is created and connected to all stations. Bandwidths of all links of a station are summed up. The delay is defined as the average delay of the links.

4.5 Graph "Clique for Stations"

Graph "Clique for Stations" is similar to the "Large clique for links" (see Subsection 4.2), only links of one

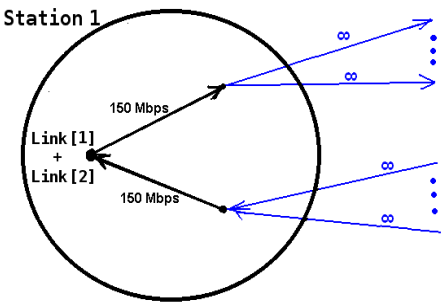

Fig. 4. Model "Clique for Stations"

station are combined into one vertex. The capacity of this vertex is defined as the sum of the bandwidths of its constituent links. To represent the throughput of a vertex through arc capacities, two additional vertices are created, one is responsible for incoming connections, the other is responsible for the outgoing (as in Subsection 4.2 above). Thus, the station model, shown in Fig. 2 is converted to the from of Fig. 4.

5. EXPERIMENTAL COMPARISON OF ALGORITHMS FOR SOLVING THE PACKET ROUTING PROBLEM

The results presented in this section show the performance of the greedy algorithm based on the Dijkstra's algorithm, compared to the FPTAS, based on the trancated Bellman-Ford algorithm.

Description and numbering of test instances:

(1) *"Clique for links"* ×2 is a graph with cliques for links, formed according to Subsection 4.2 with doubled throughput of links of base stations.
(2) *"Star for links"* is a graph formed according to Subsection 4.3.
(3) *"Star for links"* ×2 is a the graph formed according to Subsection 4.3 with doubled bandwidth of base station links.
(4) *"Star for stations"* is a graph formed according to Subsection 4.4.
(5) *"Star for stations"* ×2 is a graph formed according to Subsection 4.4 with doubled capacity of base station links.
(6) *"Satellites, random sessions"* is a graph that contains only vertices modeling the satellites (no base stations). Sessions are randomly generated in order to get high traffic.
(7) *"Clique for stations"* ×2 is a graph formed according to Subsection 4.5 with doubled bandwidth of base station links.

The source-destination pairs were generated so as to model the global telecommunication flows. We assumed that the number of active users in each square unit of the Earth surface is proportional to population on the unit. The origin and the destination of each call is chosen at random among active users. All active users are assigned to the nearest satellite or ground station.

The dimensions of the constructed packet routing problems are given in Table 1. The total amount of data transfer was 2246.3 Mbit/sec in all instances, except for

Table 1. Test problem parameters

Instance	n	m	N	k
1	543	19474	245481	12373
2	272	1292	245481	12373
3	272	1292	245481	12373
4	197	992	245481	12373
5	197	992	245481	12373
6	135	750	1517697	743
7	318	4652	245481	12373

Table 2. The estimate of relative error ω'

	greedy algorithm	FPTAS using Bellman-Ford alg.	
		$\omega = 0.2$, $L_{\max} = 13$	$\omega = 0.4$, $L_{\max} = 10$
1	0	8.5×10^{-10}	0.005
2	0	0.003	0.009
3	0.06	8.5×10^{-10}	0.005
4	0	0.004	0.01
5	0	0.0001	0.005
6	0.066	0.01	0.058
7	0	6.3×10^{-5}	0.005

Problem 6, where $\sum_{i=1}^{N} W_i = 14028$ Mbit/sec. The link bandwidth is ranging from 100 to 300 Mbit/sec. The transfer rate of each session is 9600 bit/sec which may be considered to be negligibly small compared with link capacities and source-destination transfer demands. This motivates the application of the FPTAS (with subsequent routing of all sessions according with the relaxed problem solution) and its comparison to the greedy algorithm, where each session is routed explicitly.

Values of the objective function for the heuristic solutions were compared to the LP upper bound for the optimum computed using CPLEX (see e.g. Borisovsky et al. (2019)). This allowed us to calculate the upper bound ω' on the relative error in Table 2. If the algorithm obtained a solution with objective function value f_{appr}, while the LP optimum equals f^*, then the upper bound on the relative error is $\omega' = (f^* - f_{appr})/f^*$.

In general, we can conclude that on the practical instances the FPTAS obtains solutions with approximation ratio ω' much smaller than the approximation guarantee ω. The FPTAS is more accurate than greedy algorithm in the worst case, but on some instances the greedy finds the optimal solutions, while FPTAS solutions have some error.

5.1 Delay and Number of Arcs in Routes

The delay in the transfer of sessions for the solutions found are given in Table 3. It can be seen from the table that the greedy algorithm is preferable to the FPTAS in terms of delay because it takes into account the delays of the arcs. The average delay of FPTAS with $w = 0.2, L_{\max} = 13$ was greater than that in the case of $w = 0.4, L_{\max} = 10$. The relation between the maximal delays is the opposite.

In all problems with base stations, except for problems 6 and 7, the maximum number of arcs in the routes found by the greedy algorithm, amounted to 13. This number includes fictitious arcs that model links. In problem 6, the maximum number of arcs in routes is 12, and in task 7 it is 14. The maximum number of arcs in the FPTAS solutions always equals to $L_{\max}$.

Table 3. Delay in sessions transfer

	greedy algorithm		FPTAS, $\omega = 0.2$, $L_{\max} = 13$		FPTAS, $\omega = 0.4$, $L_{\max} = 10$	
	max	avg	max	avg	max	avg
1	145	68.5	321	109.5	371	102.4
2	285	70.4	370	119.3	392	114.2
3	219	61.6	382	142	397	135.0
4	290	66.3	377	119.25	390	114.9
5	290	70.1	379	142.2	400	136.3
6	138	54.8	143	58	165	54.7
7	146	69.4	325	109.5	368	102.8

Table 4. Computing time

	greedy algorithm time (ms)		FPTAS, $\omega = 0.2$ $L_{\max} = 13$	FPTAS, $\omega = 0.4$ $L_{\max} = 10$
	1 thread	8 threads	time (sec)	time (sec)
1	1915	333	9731.7	878
2	303	188	1372.0	93.5
3	389	141	1377.5	95.2
4	136	137	929.5	68.3
5	72.6	86.7	942.7	68.9
6	954	195	40,2	2.9
7	268	109	2664.3	272.7

5.2 Computation Time

Computation time of the heuristic algorithms obtaining the described results is given in Table 4. For these calculations Xeon E5420 QuadCore 2.5 GHz was used, 8Gb RAM. For the approximation algorithms we used the compiler from Intel C++ Composer XE for Windows. This table also contains the CPU time of a parallel implementation of the greedy algorithm, where we used a specially designed version of the Dijkstra's algorithm that may be called in parallel for each root vertex $v \in V$. The experiments were run on 8-core hardware, therefore at most 8 threads of Dijkstra's algorithm were run in parallel.

The parallel version of Greedy using 8 cores is clearly the fastest one, achieving speed-ups of about 5.7 times on instance 1 ("Click for links $\times 2$') compared to the serial version. On smaller instances this speed-up vanishes due to communication cost.

In general, it can be seen from the tables, for the instance "Click for links $\times 2$", which has the largest number of vertices and arcs, all of the tested algorithms have the highest error and the longest CPU time. However, the amount of delay in routes found by the greedy algorithm on this problem turns out to be the smallest, which is due to the most detailed modelling of delays between the ground stations in this version of the problem.

6. CONCLUSION

(1) Approximate solution of the packet routing problem in software defined networks for many practical cases can be found using the proposed greedy algorithm with relatively low computing time. The computation time for most of the considered SDSN instances was less than a second.
(2) If the transfer rate of each session is negligibly small, compared to other problem input data, then the routing problem can be solved with any chosen accuracy using the FPTAS. The computation time for the considered practical instances was from tens of seconds to few thousand seconds.
(3) In practice the FPTAS obtains solutions with approximation ratio much smaller than its approximation guarantee.

REFERENCES

Tang, Z., Zhao, B., Yu, W., Feng, Z., and Wu, C. (2014). Software defined satellite networks: Benefits and challenges. *2014 IEEE Computers, Communications and IT Applications Conference*, 127–132.

Xu, S., Wang, X.W., and Huang, M. (2018). Software-defined next-generation satellite networks: Architecture, challenges, and solutions. *IEEE Access*, 6, 4027–4041. doi:10.1109/ACCESS.2018.2793237.

REFERENCES

Baier, G.: Flows with Path Restrictions. Ph.D. Dissertation, TU Berlin, Berlin (2003)

Borisovsky P., Eremeev A., Hrushev S., Teplyakov V., Vorozhtsov M. On three approaches to length-bounded maximum multicommodity flow with unit edge-lengths. Yugoslav Journal of Operations Research, Vol. 29, N 1, 93-112 (2019)

Cormen, T.H., Leiserson, C.E., Rivest, R.L., and Stein, C.: *Introduction to Algorithms,* 2nd edition, MIT Press, 2001.

Fleischer L.K. Aproximating fractional multicommodity flow independent of the number of commodities, SIAM J.Disc.Math., 13 (2000), 505–520.

Garey, M.R. and Johnson, D.S.: Computers and intractability. A guide to the theory of NP-completeness. W.H. Freeman and Company, San Francisco (1979)

Garg N., Vazirani V., Yannakakis M. Primal-dual approximation algorithms for integral flow and multicut in trees, Algorithmica, 18, 3–20 (1997)

Tang, Z., Zhao, B., Yu, W., Feng, Z., and Wu, C. (2014). Software de
ned satellite networks: Bene
ts and chal- lenges. 2014 IEEE Computers, Communications and IT Applications Conference, 127–132.

Xu, S., Wang, X.W., and Huang, M. (2018). Software- de
ned next-generation satellite networks: Architecture, challenges, and solutions. IEEE Access, 6, 4027–4041.