Application of the Ant Colony algorithm for routing in next generation programmable networks*

Stanisław Kozdrowski[1,4][0000−0001−6647−5189],
Magdalena Banaszek[1,5][],
Bartosz Jedrzejczak[1,6][],
Mateusz Żotkiewicz[2,7][0000−0002−8049−7410], and
Zbigniew Kopertowski[3,8][0000−0002−9471−6258]

[1] Computer Science Institute, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
[2] Institute of Telecommunications, Warsaw University of Technology,
Nowowiejska 15/19, 00-665 Warsaw, Poland
[3] Orange Labs Polska, Orange Polska
Obrzeżna 7, 02-691 Warszawa, Poland
[4] s.kozdrowski@elka.pw.edu.pl
[5] magdalena.banaszek.stud@pw.edu.pl
[6] bartosz.jedrzejczak.stud@pw.edu.pl
[7] mzotkiew@tele.pw.edu.pl
[8] Zbigniew.Kopertowski@orange.com

Abstract. New generation 5G technology provides mechanisms for network resources management to efficiently control dynamic bandwidth allocation and assure the Quality of Service (QoS) in terms of KPIs (Key Performance Indicators) that is important for delay or loss sensitive Internet of Things (IoT) services. To meet such application requirements, network resource management in Software Defined Networking (SDN), supported by Artificial Intelligence (AI) algorithms, comes with the solution. In our approach, we propose the solution where AI is responsible for controlling intent-based routing in the SDN network. The paper focuses on algorithms inspired by biology, i.e., the ant algorithm for selecting the best routes in a network with an appropriately defined objective function and constraints. The proposed algorithm is compared with the Mixed Integer Programming (MIP) based algorithm and a greedy algorithm. Performance of the above algorithms is tested and compared in several network topologies. The obtained results confirm that the ant colony algorithm is a viable alternative to the MIP and greedy algorithms and provide the base for further enhanced research for its effective application to programmable networks.

* The work on this paper was done in FlexNet project in EUREKA CELTIC-NEXT Cluster for next-generation communications under partial funding of The National Centre for Research and Development in Poland.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Keywords: heuristics · artificial intelligence · ant colony · internet of things · software defined networking · mixed integer programming · programmable networks.

1 Introduction

In the paper, we present a problem solution studied in the FlexNet project and related to efficient, dynamical, flexible Software Defined Networking (SDN) resource allocation for Internet of Things (IoT) applications with different Quality of Service (QoS) requirements. In the FlexNet project [8, 5], the Artificial Intelligence (AI) based developed solution is responsible for controlling the routing of intents in SDN. Typically, network is managed statically using commands and scripts, so the efficiency of resource provisioning is low and practically without automatization. Over the last few years we can observe new network solutions with improved management, where most advanced is the SDN solution [17, 28]. In IETEF RFC7149 [12] SDN is defined as a set of mechanisms and techniques used to build network services in deterministic, dynamic, and scalable methodology, suitable for use in 5G technology [24]. SDN Controller allows for adaptive, dynamic resources provisioning by applying management rules to traffic flows in the network [6]. On the other hand, also the traffic in the network becomes more complex, especially in IoT applications [2, 30, 22, 21], with big data volume generated to the network, and requires more flexibility and scalability [25, 14]. New applications and appearing different types of devices generate different patterns of network traffic. Therefore, current network solutions often do not meet the arising needs and their management is not effective. Another disadvantage of legacy networks is complex architecture in case of introducing QoS and security policies [23].

1.1 Motivation

Therefore, it is envisioned to use for future networks the SDN solutions with their capability of programmable flexible control of network resources and dynamic on demand configuration according to application requirements [27]. Such approach allows for flexible creation of new services and applications that are installed over the network controller while no changes in the actual network devices are needed.

Artificial Intelligence (AI) has seen a surge of interest in the networking community [32]. Recent contributions include data-driven flow control for wide-area networks, job scheduling, and network congestion control [18]. A particularly promising domain is the network management. Researchers have used Machine Learning (ML) to address a range of network tasks such as network resource control, routing, and traffic optimization [15, 4, 1]. In loT networks we expect network conditions to vary over time and space. Time varying conditions may be long term (seasonal) or short term resulting in a significant impact on network performance [20]. ML techniques will be developed in order to detect such changes and signal them to the SDN layer for timely action to be taken to improve the overall network performance. Another solution is a knowledge-based

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

network (KDN) which also is a next step on the path towards an implementation of a self-driving network [19]. KDN is a complementary solution for SDN that brings reasoning processes and ML techniques into the network control plane to enable autonomous and fast operation and minimization of operational costs.

1.2 AI challenges and FlexNet AI concept

In the FlexNet platform, SDN network orchestration will be supported by AI for solving selected problems of network resource control. The concept of AI application covers such capabilities as: flexible traffic control in the network, flexible adaptation to the conditions of the system, using appropriate learning algorithms based on the state-of-the-art approaches to the problem, i.e., Reinforcement Learning (RL), [31], reaction in real time. The sole goal of the AI will be to distribute network resources in the way to maximize a global objective function related to QoS, e.g., minimize buffer occupancy sizes in the nodes in order to minimize network characteristics like packet losses or packet delays.

FlexNet AI uses the concept of Off-Platform Application (OPA). The application consists of two main blocks, i.e., Path Generator and Maintainer, and communicates directly with ONOS controller [26] and their two build-in applications, i.e., Intent Forwarding (IFWD) and Intent Monitoring and Rerouting (IMR). Also, it communicates directly with switches on a network using ifstat external application [13] or sFlow-RT monitoring tool [29] (see Figure 1).

The first block of FlexNet AI, i.e., Path Generator, is responsible for generating a pair of paths for each intent being registered. The paths are selected in a way to balance efficiency and available capacity providing one fast and one spacious path.

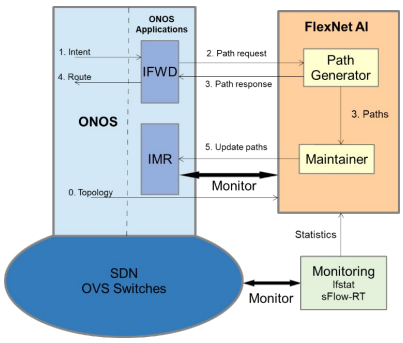


Fig. 1: Flexnet AI architecture.

The second block, i.e., Maintainer, is responsible for switching intents between paths selected by the Path Generator in a way to maximize efficiency. The data used by FlexNet AI to select paths and maintain intents consists of static and dynamic entries. In the first group, entries deal with the topology of a network. They are obtained directly from ONOS during start-up. In the second group the dynamic data are collected, constantly updated using the feedback generated by the ONOS IMR build-in application and the external ifstat application.

The first task of FlexNet AI is to route intents. The IFWD application contacts the Path Generator informing FlexNet AI that the intent is present. The Path Generator computes two paths based on the current network state and previously routed intents. One of the computed paths is selected by the Path

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Generator and sent back to the IFWD application, where it is forwarded to ONOS controller to establish a route for the intent. The main goal is to find two paths, which are edge-disjoint and have the best cost function related to QoS parameters, i.e., delay and/or loss. For this purpose, we used a nature-inspired Ant Colony (AC) algorithm [7, 10]. The novelty of our approach is to use the AC algorithm to find two paths simultaneously. In general, until now, the AC algorithms have been used to find only one path at a time. [11, 16, 3].

The article is organized as follows: in Section 2 we briefly describe the problem and algorithms we use in the contribution. Section 3 presents the results of the experimental study. Contributions of this work are summarized and future work directions are discussed in Section 4.

2 Problem Formulation and Algorithms

In this contribution the goal of the AI is to rationally control the routes of intentions in an SDN network. To this end, for each intention we choose a pair of potential paths that are disjoint as much as possible. This pair is computed in a way that minimises the weighted average cost of the paths by assuming that the cost of one path is given by its length and the cost of the other by its occupancy. In this way, we obtain a pair of paths, one of which minimises the transmission time and the other minimises the probability of loss. We find the sought pair using mainly the Ant Colony (AC) algorithm. Our AC algorithm is designed specifically for this problem and is compared to commonly known algorithms (MIP, Mixed Integer Programming) that guarantee an optimal solution and to greedy algorithms.

2.1 Problem formulation

The problem we are considering in this article belongs to the class of NP-complete problems [9] and is presented as follows:

Data Input data consist of the following items:

- demand between s and d nodes,
- network with capacities,
- actual traffic on arcs.

They can be formally presented using the following sets and constants.

Sets

$\mathcal{V}$ vertices
$\mathcal{A}$ arcs
$\delta^+(v)$ set of arcs entering vertex $v \in \mathcal{V}$
$\delta^-(v)$ set of arcs leaving vertex $v \in \mathcal{V}$
$\mathcal{I}$ set of possible lengths of the first path; $|\mathcal{I}| = |\mathcal{A}|$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Constants

s source; $s \in \mathcal{V}$
d destination; $d \in \mathcal{V}$
$b(a)$ used fraction of bandwidth on arc $a \in \mathcal{A}$
ξ' weight of the first path
ξ'' weight of the second path
L_X maximum tolerable used fraction of bandwidth for the first path
L_Y maximum tolerable used fraction of bandwidth for the second path

Assumptions In the problem, we assume that the load on an arc is the actual traffic on it. Moreover, the load on an arc cannot be smaller than 0.01 to prevent forming loops on the second path.

Objective and formal model We search for the pair of paths X and Y between s and d such that:

- the number of common arcs is minimal (highest priority); in other words, we are interested in disjoint paths if they exist; if there is at least one solution without common arcs, we choose and return one of them; if there is not, and there is a solution with one common arc, we return it, and so forth.
- path X does not use arcs with load greater than L_X, path Y does not use arcs with load greater than L_Y; these constraints must always be satisfied; if there are no paths satisfying the constraints, then there is no solution.
- maximise a weighted sum $\xi' A + \xi'' B$, where A is the reciprocal of the length of path X and B is the product of free capacities on path Y (if Y is completely free then $B = 1$; if Y consists of two arcs that are occupied in half, then $B = 0.25$); in other words, the best solution is when X is the shortest and Y is not occupied in one percent (see Assumptions above).

The problem can be formally presented using the following set of variables and constraints.

Variables

x_a binary; 1 if arc $a \in \mathcal{A}$ belongs to X
y_a binary; 1 if arc $a \in \mathcal{A}$ belongs to Y
z_a binary; 1 if arc $a \in \mathcal{A}$ is used by both X and Y
T integer; number of common arcs

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Objective and constraints

$$\min\left\{(\xi'+\xi'')T \;-\; \frac{\xi'}{\sum_{a\in\mathcal{A}} x_a} \;-\; \xi'' \prod_{a\in\mathcal{A}:y_a=1}(1-b(a))\right\} \tag{1a}$$

$$\sum_{a\in\delta^+(v)} x_a - \sum_{a\in\delta^-(v)} x_a = \begin{cases} 1 & v = d \\ 0 & v \in \mathcal{V}\setminus\{s,d\} \end{cases} \tag{1b}$$

$$\sum_{a\in\delta^+(v)} y_a - \sum_{a\in\delta^-(v)} y_a = \begin{cases} 1 & v = d \\ 0 & v \in \mathcal{V}\setminus\{s,d\} \end{cases} \tag{1c}$$

$$x_a = 0 \quad \forall a \in \mathcal{A} : b(a) > L_X \tag{1d}$$

$$y_a = 0 \quad \forall a \in \mathcal{A} : b(a) > L_Y \tag{1e}$$

$$x_a + y_a \le 1 + z_a \quad \forall a \in \mathcal{A} \tag{1f}$$

$$\sum_{a\in\mathcal{A}} z_a \le T \tag{1g}$$

Objective function (1a) consists of three elements. The first one, i.e., $(\xi' + \xi'')T$ is responsible for the priority objective, which is minimizing the number of common arcs. We will call this part of the objective function the First Criterion in Numerical Results. The second element represents the cost of path X and the third element represents the cost of path Y. They will be together called the Second Criterion in Numerical Results. Notice that the second element cannot exceed $-\xi'$ and the third element cannot reach $-\xi''$ (load cannot be smaller than 0.01), thus each change in T (First Criterion) has the absolute priority over other changes (Second Criterion).

Constraints (1b) and (1c) impose the flow conservation law on paths X and Y, respectively, while (1d) and (1e) assure that the paths do not use overloaded arcs. Finally, (1f) and (1g) set the correct number of shared arcs.

2.2 Algorithms

MIP approach The presented problem is not-linear and cannot be directly solved using MIP solvers. In this section, we present an algorithm that using a linear version of a simplified problem solves the considered problem invoking an MIP solver a number of times. In the simplified model, the following additional constants are used.

S length of the shortest path between s and d
P maximum number of arcs in X
T maximum number of shared arcs (variable in the base model).

The approach is presented in Algorithm 1. It is a brute force approach that checks possible values of pairs T and P. For each value T it first checks the feasibility of solutions setting P to the maximum possible value. If a solution cannot be found, the next value for T is considered. The approach uses a simplified MIP

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Algorithm 1: MIP

```
Input: S = shortestPath(weight(a) = 1)
best_Y = −cost(shortestPath(weight(a) = −log(1 − b(a))))
for T=0,1,... do
    best_obj = 0;
    for P=|V| − 1,S,S+1,...,|V| − 2 do
        if best_obj ≥ ξ′/P + ξ″e^{best_Y} then
            return best_res;
        end
        obj, res = solve simplified MIP;
        if obj=NULL and P=|V| − 1 then
            break
        end
        if obj then
            real_obj = ξ′/P + ξ″e^{obj};
            if real_obj > best_obj then
                best_obj = real_obj;
                best_res = res;
            end
        end
    end
    if best_res then
        return best_res;
    end
end
```

model with modified objective function and an additional constraint shown in (2).

$$\max \sum_{a \in \mathcal{A}: b(a) \leq L_Y} y_a \log(1 - b(a)) \tag{2a}$$

$$\sum_{a \in \mathcal{A}} x_a \leq P \tag{2b}$$

Constraint (2b) assures that the length of path X in the obtained solution will not exceed P. Having constant T and iterating through all viable values of P, the objective function (1a) reduces to the third element, which is the product of fractions of free capacities on arcs used by path Y. The product can be made linear using a logarithm, which is visible in (2a).

Ant-colony approach Algorithm 2 is based on behaviour of an ant colony. In each iteration every ant is moved by one arc from source to destination (forward) or backward, when the destination was already reached. When moving forward ant chooses an arc depending on the pheromone amount. The more pheromone lays on an arc, the more attractive it is for an ant. With some small probability an ant can choose a less attractive arc. An ant cannot use arcs with load exceeding L_X. Each ant remembers nodes visited on the path and avoids choosing arcs that could create a loop. In result, paths determined by ants fulfill requirements of path X, which should be the shortest and not overloaded. When an individual reaches the destination node, the Dijkstra algorithm finds path Y. It has the

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Algorithm 2: Ant colony

```
Input: Lx = 0.9
Input: Ly = 0.99
Input: n = number of iterations
best_Y = −cost(shortestPath(weight(a) = −log(1 − b(a))))
for i=0,...n do
    for ant in ant_colony do
        if ant.moving_forward then
            if ant.current_node == destination then
                x_path = ant.found_path
                y_path = Dijkstra(x_path)
                obj = Σ_{y∈y_path} log(1 − b(y))
                ant.evaluation = ξ′/len(x_path) + ξ″e^{obj}
            else
                arc = choose_arc_by_pheromone()
                move_ant(ant, arc)
            end
        else
            arc = pop(ant.found_path)
            move_ant(ant, arc)
            leave_pheromone(arc)
        end
    end
    if i % evaporation_frequency == 0 then
        evaporate(pheromone_decrement, arcs)
    end
end
return find_best_solution(ant_colony)
```

minimal number of common arcs with path X, because we explicitly set weights of arcs belonging to X found by an ant to a sufficiently great number. Arcs with load exceeding L_Y cannot be included into path Y. The selected pair of paths is evaluated by calculating weighted sum, which influences the amount of pheromone that will be left on arcs while moving backward. When specified number of iterations elapses, some of the pheromone evaporates from each arc. After the final iteration the last pairs of paths remembered by ants are compared. A pair with the highest evaluation value is returned as the solution.

Greedy approach Algorithm 3 presents the greedy approach based on the Dijkstra algorithm. At first it searches for path X, choosing the shortest path consisting of arcs with load less than L_X. The Dijkstra algorithm minimises a path cost, where all arcs have weight equal to 1. When choosing a neighbour to visit, arcs with the load exceeding L_X are not considered. Then, also using the Dijkstra algorithm, path Y is determined based on the chosen path X. All arcs that belong to previously found path X have much bigger weight than others that have the weight reflecting the load. Moreover, arcs with load exceeding L_Y cannot be used by path Y. The algorithm minimises the path cost, which results in finding path Y that has the least common arcs with path X and has the most free capacity on its arcs. The found pair of paths is evaluated with the weighted sum and returned as a solution.

We will refer to this approach as the DijkstraXY algorithm. Another greedy approach that we use in the research reverses the path selection order picking

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

the optimal path Y at first and then selecting path X which is the shortest and as disjoint as possible from path Y. The latter approach will be called the DijkstraYX algorithm.

Algorithm 3: Greedy algorithm

Input: $Lx = 0.9$
Input: $Ly = 0.99$
$best\_Y = -cost(shortestPath(weight(a) = -log(1 - b(a)))$
x_path = Dijkstra()
y_path = Dijkstra(x_path)
$obj = \sum_{y \in y\_path} \log(1 - b(y))$
x_y_evaluation = $\xi'/len(x\_path) + \xi'' e^{obj}$
return x_path, y_path

3 Experiments and Results

The common configuration parameters for AC algorithm for all simulations studies are listed in Table 1. Objective weight ξ' was set to the length of the shortest path between s and d. This way the part of the objective function that deals with the first path is scaled. The second part of the objective was also scaled by setting ξ'' to the reciprocal of e to the power of the inverse of the length of the shortest path between s and d taking $log(1 - b(a))$ as arc weights. In addition, ξ'' was multiplied by 4. In this way, we obtained a set of problems where the second path is slightly prioritized over the first path in the constant extend.

Table 1: Parameters used in AC algorithm.

Name	Short description	Value
S	$s - d$ distance	variable
m	size of the colony	$2\|V\|$
n	number of iterations	$40S$
α	pheromone dosage factor	5
β	pheromone evaporation factor	10
γ	pheromone evaporation frequency	$2S$

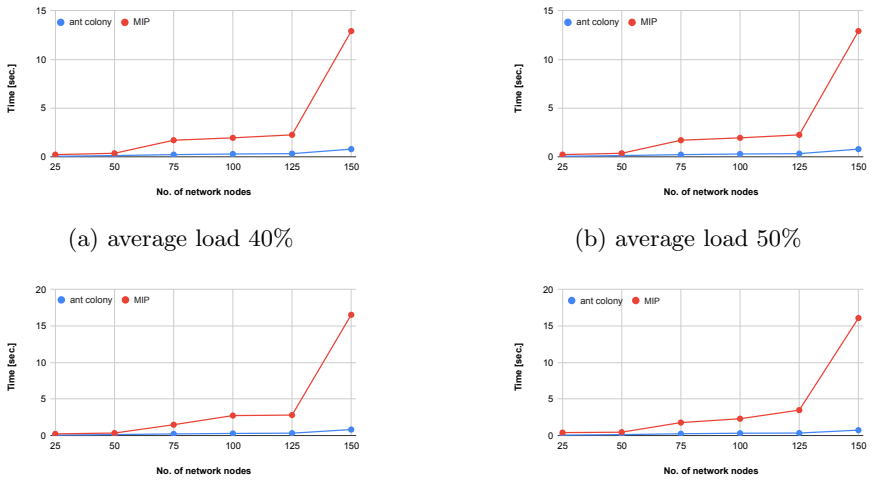
The calculations were carried out for all considered algorithms on a 2.1 GHz Xeon E7-4830 v.3 processor with 256 GB RAM running under Linux Debian operating system and additionally for MIP-based algorithm a linear solver engine of CPLEX 12.8.0.0 was used.

Each point in each graph presented in Figures 2-4 (i.e. one network with a certain load level) represents the average result for 100 instances, among which there are 10 different load distributions. For each instance, different source and

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

destination nodes are drawn uniformly at random. Each load distribution was obtained by constantly generating random demands and routing them in a network using shortest paths. Demands that could not have been satisfied were discarded. The process was repeated until the requested average load was reached. The AC algorithm, as a non-deterministic algorithm, was run for each instance 10 times and the result was averaged. On the other hand, the Dijkstra algorithm and MIP-based algorithm, as deterministic algorithms, were run one time each.

In Figure 2, running times of the presented algorithms are displayed. We can observe the MIP-based algorithm having problems with finding optimal solutions in the time limit acceptable for dynamic environments presented in the introduction. The running times can be annoying in networks of 75 nodes and more. In networks of 150 nodes and more, the running times become unacceptable for practical implementations of the presented SDN framework. On the other hand, other presented algorithms were able to solve the problem in the assumed time limit of one second. Notice that the greedy algorithms are not depicted in the figure. They running times never exceeded 0.01 second.



(a) average load 40%

(b) average load 50%

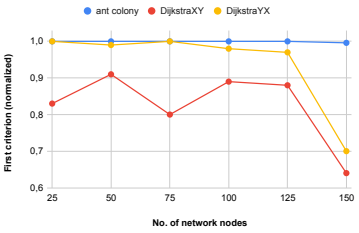
(c) average load 60%

(d) average load 70%

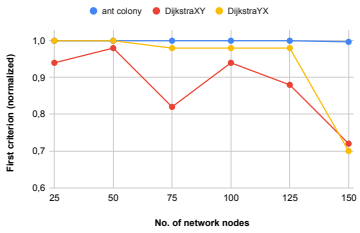
Fig. 2: Comparison of calculation times for different methods and different network loads: 40% - (a), 50% - (b), 60% - (c) and 70% - (d).

In Figure 3, the results for the First Criterion are displayed. The presented greedy algorithms were not able to stand against the MIP-based approach and the Ant-Colony approach returning results with significantly more common links. The trend is clearly visible when the number of nodes reached a certain level, which in our experiments was close to 150. It is worth to notice that the Ant-

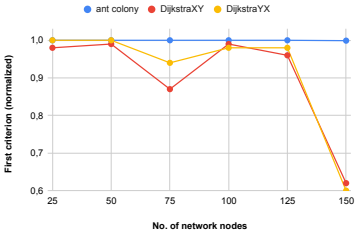
ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40



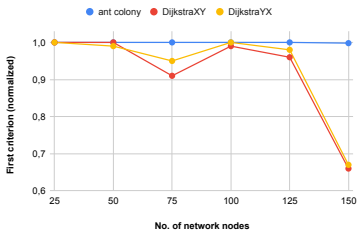
(a) average load 40%



(b) average load 50%

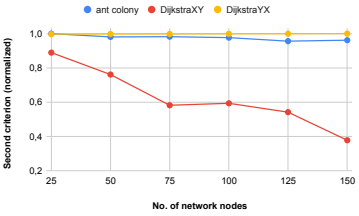


(c) average load 60%

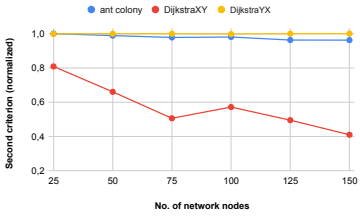


(d) average load 70%

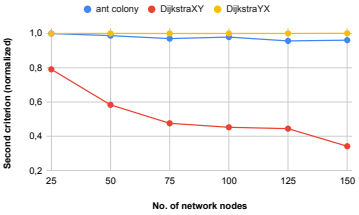
Fig. 3: Normalized objective function presenting relative number of common links (First Criterion) with respect to MIP-based method for different network loads: 40% - (a), 50% - (b), 60% - (c) and 70% - (d).



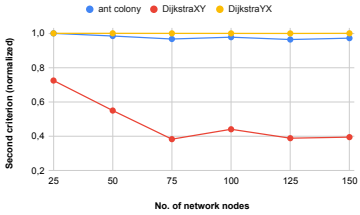
(a) average load 40%



(b) average load 50%



(c) average load 60%



(d) average load 70%

Fig. 4: Normalized objective function presenting Second Criterion with respect to MIP for different net. loads: 40% - (a), 50% - (b), 60% - (c) and 70% - (d).

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

Colony approach does not have serious problems with returning optimal solutions with respect to the First Criterion for any of the considered network instances.

Finally, in Figure 4, the obtained results for the Second Criterion are displayed. Here it is important to explain the way the figure was prepared. Because the Second Criterion counts only when the First Criterion is optimal, we decided to include only such cases in this figure. With respect to the Second Criterion, the greedy DijkstraXY algorithm is heavily outperformed by other approaches. On the other hand, DijkstraYX algorithm always finds solutions with optimal values of the Second Criterions, thus slightly outperforming the Ant-Colony approach. However, it results from the fact that the DijkstraYX algorithm was strictly designed to optimize the second (more important, due to $4\xi' = \xi''$) part of the Second Criterion. In other words, DijkstraYX returns optimal solutions with respect to the Second Criterion, but seldom finds optimal solutions with respect to the First Criterion. On the other hand, the Ant-Colony approach repeatedly finds optimal solutions with respect to the First Criterion, which occasionally results in slightly worse performance with respect to the Second Criterion.

4 Conclusions

The paper studied the problem of traffic routing in mesh networks with a specific criteria of the objective function. The presented solution is applicable to intelligent SDN management systems with AI support, especially in programmable next generation networks (5G and beyond). Moreover, the network resource allocation in context of QoS assurance is the arising problem especially for IoT services. The designed solution based on the AC algorithm is suitable for the fast and optimal resource allocation, especially in the case of emergency and delay sensitive IoT services. In FlexNet project the solution is tested for the emergency surveillance video IoT service.

A heuristic method inspired by biology called the AC algorithm is proposed to solve the problem. The novel approach of this method is the optimal routing of a pair of paths in the network, searched simultaneously, considering three criteria described in the objective function. The considered problem has been modelled and solved as an MIP problem. Thus, there is a certainty of finding an optimal solution. We demonstrated the stability of AC through simulations. We showed that it quickly converges to the best path under situations when traffic characteristics change (among others when load on the network is increased). We have shown that with an appropriate tuning of the parameters, AC behaves better when compared to other competing approaches in mesh networks. In addition, the promising results shown in the paper underline the need for a real-life testbed evaluation on which we are currently working in FlexNet project.

Further research will be focused on more comprehensive experiments that include real scenarios and with comparisons to other competitive heuristics.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

References

1. Abar, T., Letaifa, A., El Asmi, S.: Machine learning based qoe prediction in sdn networks. pp. 1395–1400 (06 2017). https://doi.org/10.1109/IWCMC.2017.7986488
2. Bera, S., Misra, S., Vasilakos, A.V.: Software-defined networking for internet of things: A survey. IEEE Internet of Things Journal **4**(6), 1994–2008 (2017). https://doi.org/10.1109/JIOT.2017.2746186
3. Bokhari, F.S., Záruba, G.V.: On the use of smart ants for efficient routing in wireless mesh networks. CoRR **abs/1209.0550** (2012), http://arxiv.org/abs/1209.0550
4. Chen, B., Wan, J., Lan, Y., Imran, M., Li, D., Guizani, N.: Improving cognitive ability of edge intelligent iiot through machine learning. IEEE Network **33**(5), 61–67 (2019). https://doi.org/10.1109/MNET.001.1800505
5. Choque, J., Agüero, R., Kopertowski, Z., Nguyen, K.K., Medela, A., Municio, E., Marquez-Barja, J.M., Domaszewicz, J., Bak, A., Lee, J.H., Noh, S., Muñoz, L.: Flexnet: Flexible networks for iot based services. In: 2020 23rd International Symposium on Wireless Personal Multimedia Communications (WPMC). pp. 1–6 (2020). https://doi.org/10.1109/WPMC50192.2020.9309486
6. Dinh, K.T., Kukliński, S., Osiński, T., Wytrebowicz, J.: Heuristic traffic engineering for sdn. Journal of Information and Telecommunication **4**(3), 251–266 (2020). https://doi.org/10.1080/24751839.2020.1755528
7. Dobrijevic, O., Santl, M., Matijasevic, M.: Ant colony optimization for qoe-centric flow routing in software-defined networks. In: 2015 11th International Conference on Network and Service Management (CNSM). pp. 274–278 (2015). https://doi.org/10.1109/CNSM.2015.7367371
8. Flexnet: Flexible iot networks for value creators. (2020), https://www.celticnext.eu/project-flexnet/
9. Garey, M.R., Johnson, D.S.: Computers and Intractability; A Guide to the Theory of NP-Completeness. W. H. Freeman & Co. New York, NY, USA ©1990 (1990)
10. Guan, Y., Gao, M., Bai, Y.: Double-ant colony based uav path planning algorithm. In: Proceedings of the 2019 11th International Conference on Machine Learning and Computing. p. 258–262. ICMLC '19, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3318299.3318376
11. Hamrioui, S., Lorenz, P.: Bio inspired routing algorithm and efficient communications within iot. IEEE Network **31**(5), 74–79 (2017). https://doi.org/10.1109/MNET.2017.1600282
12. IETF: Software-defined networking: A perspective from within a service provider environment. (2017), https://tools.ietf.org/html/rfc7149
13. ifstat: ifstat - linux man page. (2017), https://linux.die.net/man/1/ifstat
14. Jin, Y., Gormus, S., Kulkarni, P., Sooriyabandara, M.: Content centric routing in iot networks and its integration in rpl. Comput. Commun. **89**(C), 87–104 (Sep 2016). https://doi.org/10.1016/j.comcom.2016.03.005
15. Kozdrowski, S., Cichosz, P., Paziewski, P., Sujecki, S.: Machine learning algorithms for prediction of the quality of transmission in optical networks. Entropy (Basel, Switzerland) **23**(1) (January 2021). https://doi.org/10.3390/e23010007
16. Liu, X., Li, S., Wang, M.: An ant colony based routing algorithm for wireless sensor network. International Journal of Future Generation Communication and Networking **9**, 75–86 (06 2016). https://doi.org/10.14257/ijfgcn.2016.9.6.08
17. Liyanage, M., Ylianttila, M., Gurtov, A.: Securing the control channel of software-defined mobile networks (06 2014). https://doi.org/10.1109/WoWMoM.2014.6918981

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40

18. Mao, H., Alizadeh, M., Menache, I., Kandula, S.: Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. p. 50–56. HotNets '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/3005745.3005750, https://doi.org/10.1145/3005745.3005750
19. Mestres, A., Rodriguez-Natal, A., Carner, J., Barlet-Ros, P., Alarcón, E., Solé, M., Muntés-Mulero, V., Meyer, D., Barkai, S., Hibbett, M.J., Estrada, G., Ma'ruf, K., Coras, F., Ermagan, V., Latapie, H., Cassar, C., Evans, J., Maino, F., Walrand, J., Cabellos, A.: Knowledge-defined networking. SIGCOMM Comput. Commun. Rev. **47**(3), 2–10 (Sep 2017). https://doi.org/10.1145/3138808.3138810
20. Mishra, P., Puthal, D., Tiwary, M., Mohanty, S.P.: Software defined iot systems: Properties, state of the art, and future research. IEEE Wireless Communications **26**(6), 64–71 (2019). https://doi.org/10.1109/MWC.001.1900083
21. Municio, E., Latré, S., Marquez-Barja, J.M.: Extending network programmability to the things overlay using distributed industrial iot protocols. IEEE Transactions on Industrial Informatics **17**(1), 251–259 (2021). https://doi.org/10.1109/TII.2020.2972613
22. Municio, E., Marquez-Barja, J., Latré, S., Vissicchio, S.: Whisper: Programmable and flexible control on industrial iot networks. Sensors **18**(11) (2018). https://doi.org/10.3390/s18114048
23. Murat Karakus, A.D.: "Quality of service in software defined networking: A survey". Journal of Network and Computer Applications, **Volume 80**, pages 200–218, (February, 2017). https://doi.org/10.1016/j.jnca.2016.12.019
24. de la Oliva, A., Li, X., Costa-Pérez, X., Bernardos, C., Bertin, P., Iovanna, P., Deiß, T., Mangues-Bafalluy, J., Mourad, A., Casetti, C., Gonzalez, J., Azcorra, A.: 5g-transformer: Slicing and orchestrating transport networks for industry verticals. IEEE Communications Magazine **56**, 78–84 (08 2018). https://doi.org/10.1109/MCOM.2018.1700990
25. Omar, H.: Intelligent traffic information system based on integration of internet of things and agent technology. International Journal of Advanced Computer Science and Applications **6** (02 2015). https://doi.org/10.14569/IJACSA.2015.060206
26. ONOS: ONOS Project. (2017), https://wiki.onosproject.org/
27. Open Networking Foundation: "Software-Defined Networking: The new norm for networks". White Paper, (2012)
28. Rothenberg, C.E., Nascimento, M.R., Salvador, M.R., Corrêa, C.N.A., Cunha de Lucena, S., Raszuk, R.: Revisiting routing control platforms with the eyes and muscles of software-defined networking. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks. p. 13–18. HotSDN '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2342441.2342445
29. sFlow: sflow-rt documentation. (2017), https://sflow-rt.com/reference.php
30. Thubert, P., Palattella, M., Engel, T.: 6tisch centralized scheduling: When sdn meet iot (10 2015). https://doi.org/10.1109/CSCN.2015.7390418
31. Yao, H., Mai, T., Xu, X., Zhang, P., Li, M., Liu, Y.: Networkai: An intelligent network architecture for self-learning control strategies in software defined networks. IEEE Internet of Things Journal **5**(6), 4319–4327 (2018). https://doi.org/10.1109/JIOT.2018.2859480
32. Zhao, Y., Le, Y., Zhang, X., Geng, G., Zhang, W., Sun, Y.: "A survey of networking applications applying the software defined networking concept based on machine learning". IEEE Access, (July, 2019). https://doi.org/10.1109/ACCESS.2019.2928564

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77970-2_40