Ant Colony Optimization Algorithm for Optimal Network Path Selection and Deployment of Emulated Branched Test Environments

Oleksandra Yaroshevska, Veronika Kirova, Eduard Siemens

Hochschule Anhalt, FB Elektrotechnik, Maschinenbau und Wirtschaftsingenieurwesen

Abstract

This article shows the possibilities of using the Ant Colony Optimization algorithm to find the shortest path in the communication networks. The Mininet emulator is studied as a platform for testing the algorithm. The issue of loops in branched Mininet networks due to limitations in the design of the network emulator and possible risks in their occurrence is considered. A solution to this limitation using an external controller is also proposed.

1. Introduction

1.1 Algorithm choice for finding the shortest path in a branched network

As modern technologies advance and the demand for network infrastructure increases, the challenge of establishing numerous branched local and global networks becomes more pressing. One crucial aspect of ensuring network efficiency lies in finding optimal routes between nodes.

Traditional methods like Dijkstra's algorithm [1] and the Floyd-Warshall algorithm [2] deliver precise results but often demand substantial computational resources, particularly when dealing with expansive networks. Moreover, they may not always accommodate dynamic network changes such as temporal delays or node loads. Given these limitations, there is a necessity for metaheuristic approaches to solve the problem of identifying optimal network paths. Such methods offer a flexible and adaptive approach to problem-solving, rendering them well-suited for the dynamic and intricate nature of contemporary networks [3]. One such method is the Ant Colony Optimization algorithm, inspired by ants' behavior when seeking the shortest path to a food source.

ACO presents a novel approach to optimizing paths in a network by modeling ants' decision-making process, where ants deposit pheromones on graph edges based on the quality of the path they traverse (level of delay between specific nodes, throughput levels, or other parameters, which are critical in a given network). This enables the algorithm to seek globally optimal paths while considering local information about path quality [4]. One of ACO's advantages lies in its ability to adapt to dynamic network changes and find approximately optimal solutions within a reasonable timeframe. Furthermore, ACO can be effectively parallelized: “ants” can simultaneously operate on different network segments or on various computers within the network environment. Consequently, parallel “ants” can conduct independent iterations of pheromone updates, reducing the time required to find the optimal solution and making the ACO algorithm appealing for deployment in modern computing environments.

1.2 Description of the ACO algorithm and research potential

Over the years of research, numerous studies have been conducted on the application of ACO in various fields, such as network routing, optimization of production processes, combinatorial optimization problems, and others [5, 6]. Despite extensive research, ACO still holds potential for further development and improvement. Particularly, there is potential in utilizing ACO in large local and wide area networks to address various optimization problems and improve performance. The application of ACO can lead to enhanced efficiency and reliability of networks, as well as reduced costs for their maintenance and operation.

To implement the algorithm, the Python programming language was chosen as one of the most effective tools for developing a shortest path search algorithm, as well as for other related tasks in graph analysis. Its convenient and readable syntax, along with a rich library of third-party modules provides a wide range of functionality for working with graphs and implementing various algorithms.

A diagram illustrating the operation of the ACO algorithm step by step is presented in Figure 1.

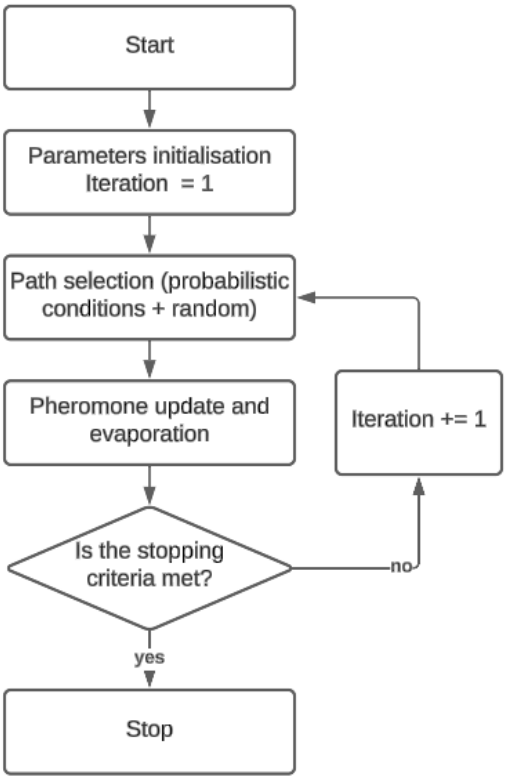

Figure 1: The operation diagram of the ACO algorithm

The elaborated algorithm may find application in further research. Particularly, one of the promising directions is the study of Application Layer Multicast (ALM), which involves the use of specialized protocols and algorithms to organize efficient data transmission among multiple network participants [7]. Unlike traditional Multicast methods, ALM allows for more flexible data flow management and ensures a higher level of reliability and quality of service. Moreover, ALM opens up new opportunities for developing distributed real-time applications such as video conferencing, online gaming, and video streaming, enabling efficient data transmission even with a large number of participants and under varying network conditions.

2. Experimental topology

2.1 Overview and choosing the testbed platform

Mininet was chosen as the testing environment for the ACO algorithm. Mininet is a tool for creating virtual network environments widely used in computer network research and development [8]. It allows for the creation of scalable and flexible virtual network topologies on a single computer or cluster of computers. Researchers and developers can test and debug various network protocols, routing algorithms, network applications, and services in a controlled and isolated environment provided by Mininet without the need for physical hardware. Virtual network devices created using Mininet can emulate the operation of real network devices, enabling experimentation and testing under conditions similar to real network scenarios.

Other environments for testing graph algorithms include solutions such as NS-3 and OMNeT++, as well as Packet Tracer, which specializes in Cisco devices. However, Mininet offers several advantages over these programs: it provides greater flexibility in configuring network topologies and parameters, facilitating the creation of diverse network scenarios for testing. Additionally, Mininet is closely integrated with the Python programming language, simplifying automation and testing using a wide range of Python libraries and tools. While NS-3 and OMNeT++ also support Python, their integration depth and simplicity are significantly lower. Moreover, Mininet boasts faster simulation speed compared to NS-3 and OMNeT++, particularly for simulating small and medium-sized networks, which can be crucial for experimental work and algorithm testing.

2.2 The loop problem in the Mininet environment

The formation of loops in path finding and route computation in computer networks is an inherent and highly significant problem that can have serious negative consequences for the entire network infrastructure. Loops, occurring in network topologies, are characterized by a closed loop of data transmission between network devices such as switches and routers. Loops in networks lead to adverse effects such as increased traffic, redundant data copies, and node overload. They can cause unpredictable network behavior, including performance degradation, data transfer delays, and even complete network infrastructure failure.

3. Proposed solution

3.1 The loop problem solution

In the case of Mininet, the loop problem cannot be resolved using the built-in controller located within the platform, which performs the routing function in simple networks. For this reason, the decision was made to use an external controller. One of the most effective solutions in this domain is the Floodlight controller [9]. It serves as a tool for regulating data flows in the network and offers several advantages such as open-source nature, scalability, extensibility, reliability, and community support. It automatically constructs a graphical network interface and facilitates monitoring of node interactions. Furthermore, there is a pre-built Python library that provides an API for interacting with the Floodlight. This enables data flow management, routing rule configuration, network status monitoring, and more [10]. Therefore, the employment of the Floodlight controller stands as an efficient means of managing and optimizing data flows in networks.

Another solution to this issue is the Spanning Tree Protocol [11], which is also utilized for preventing loops in Ethernet networks. However, the external controller has certain advantages over STP, which led to its selection in this study. Firstly, STP blocks some ports to prevent network loops, potentially leading to inefficient bandwidth utilization and limited device accessibility. Conversely, an external controller enables traffic control without port blocking, thereby enhancing network performance. Secondly, the external controller furnishes centralized management of the entire network, simplifying configuration, monitoring, and resource management. Lastly, the controller allows for dynamic responses to network changes and real-time routing reconfiguration.

3.2 Description of the Floodlight controller operation

Consider a branched local network created using the Mininet emulator as a testbed for network transmission over the ACO algorithm's chosen path (Figure 2).

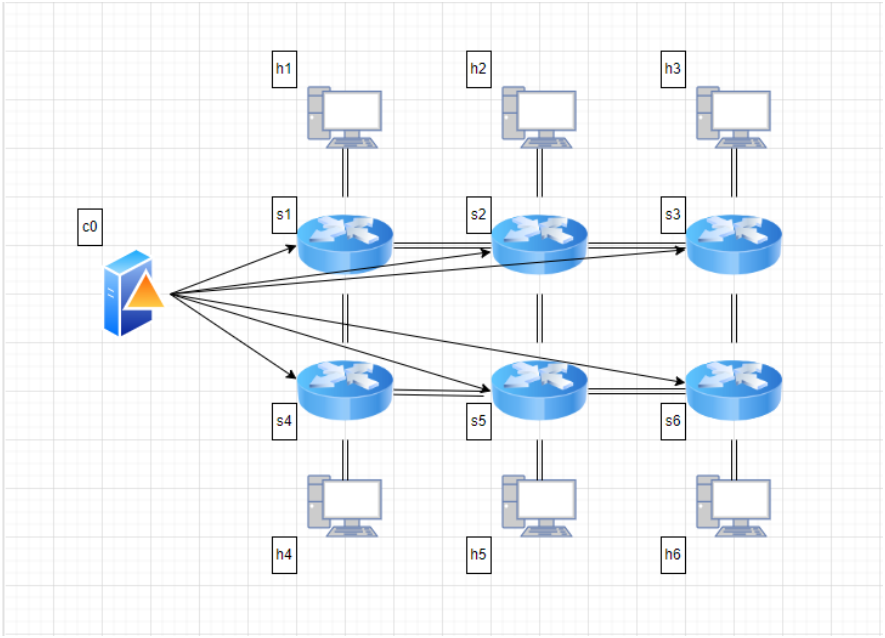

Figure 2: An extensive "6-6" network

This Figure depicts a network comprising six switches and six hosts, along with a controller that manages the data flows entering the switches. Each host is an individual Linux-based device with full capabilities to test packet transmission to other devices. In the presented diagram, only one host is connected to each switch. The network can be expanded to include any number of hosts, but this does not significantly impact solving the network loop problem.

Let's consider the transmission of data streams from host1 to host6. On the Figure 3 we have the following possible paths:

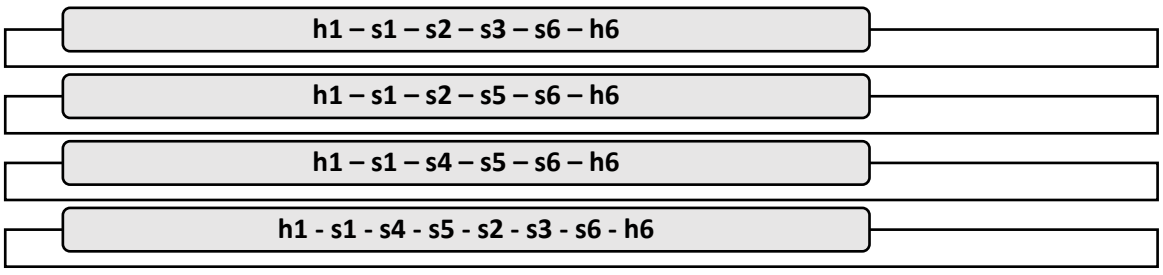

Figure 3: The possible network paths

Among these paths, the most effective one is considered to be the one with the lowest cumulative delay, highest throughput, or other parameter deemed most significant. During the operation of the ACO algorithm, this parameter can be altered dynamically, affecting the behavior of the "ants". Cyclic paths, such as "h1 - s1 - s2 - s5 - s4 - s1...", are excluded due to the closed nature of a section of the path. These paths will be automatically filtered out by the ACO algorithm and, consequently, will not be incorporated into the data transmission rules implemented via the controller.

3.3 Data Flow Management Rules

During the creation and testing of the Mininet environment, data transfer rules will be adjusted depending on the task at hand. In general, the rules for the Floodlight controller consist of the following components (Figure 4):

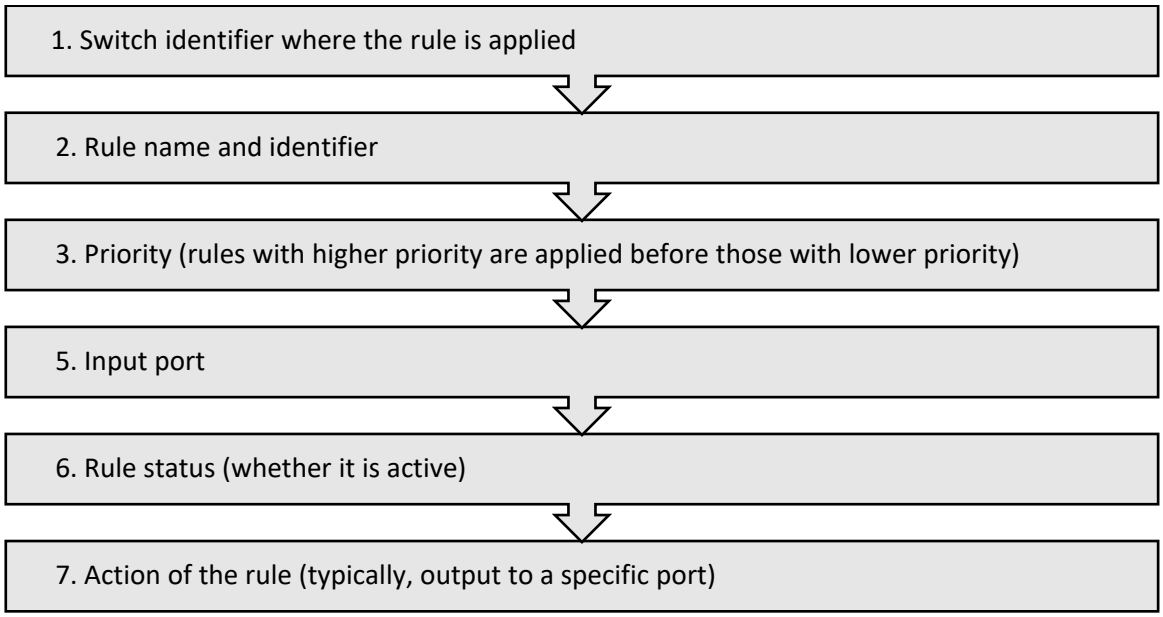

Figure 4: Rules for the Floodlight controller

Thus, by configuring rules in the controller, it becomes possible to promptly respond to network changes and tailor it to meet specific requirements. This enhances network flexibility and its ability to adapt swiftly to evolving conditions and demands.

Another crucial aspect regarding the necessity for rules in the Floodlight controller is their role in enhancing network performance. Without considering these rules, the data flow paths may not be optimal. Utilizing the ACO algorithm enables the identification of optimal paths, followed by the correct configuration of rules for the controller. Consequently, this optimization improves network performance by reducing data loss and packet transmission delays.

4. Conclusion and future work

During this study, the ACO algorithm was explored as one of the potential solutions to the problem of finding the shortest path in branched networks. This algorithm simulates the decision-making process of ants, which leave pheromones on the edges of the graph and, over time, model the most efficient path. ACO has the capability to adapt to changes in the network and explore it in parallel, which can improve the efficiency of both local and wide area networks.

Mininet was chosen as the testing environment for this task. It simulates real physical devices, including the Linux operating system on each host, and facilitates data transmission akin to a real network. While working in Mininet, the issue of loop formation in branched networks was identified and investigated, which the built-in system controller is unable to resolve. A solution was proposed based on an external Floodlight controller. This controller enables effective operation with networks where data flows between two nodes can traverse multiple paths, potentially forming loops. The proper elimination of loops is achieved through specialized flow redistribution rules, which can adapt flexibly based on the key parameter of connection efficiency between

network nodes. The proposed solution allows to provide a stable, close to real conditions testbed for testing the ACO algorithm, allowing to dynamically change its configuration depending on the experiment task.

A comprehensive literature analysis regarding path optimization problems in networks underscores the potential of ACO in enhancing the efficiency of network infrastructure, alongside opportunities for refining the algorithm to improve the accuracy of pathfinding according to predefined parameters. This research aims to further advance this method and its utilization in contemporary network scenarios. Additionally, investigating the potential of optimizing network infrastructure using combined approaches, including integrating ACO with other algorithms and technologies like machine learning and artificial intelligence, holds promise. Further application of the finalized algorithm is anticipated in the development of ALM, where each session's creation will necessitate determining the most effective data transmission path between network nodes.

5. Acknowledgement

This work has been funded by DAAD, BMBF Foundation for partnership between scholars and scientists from Ukraine and Germany within the project Fit4Ukraine. The authors thank Hochschule Anhalt and Future Internet Lab Anhalt for support and the opportunity to use the equipment for research.

6. Bibliography

[1] Likaj, Rame & Bajrami, Xhevahir & Hoxha, Gezim & Shala, Erjon (2024): "The Application of the Dijkstra Algorithm in the Finding of the Optimal Solution for the Connected Road Network to Center Prishtina", in: Proceedings of the Joint International Conference: 10th Textile Conference and 4th Conference on Engineering and Entrepreneurship, pp. 86-97.

[2] Dermawan, Tri (2019): "Comparison of Dijkstra dan Floyd-Warshall Algorithm to Determine the Best Route of Train", in: IJID (International Journal on Informatics for Development), vol. 7, No. 2, pp. 8-13.

[3] Liang, Jun-Chao & Gong, Yue-Jiao & Wu, Xiao-Kun & Li, Yuan (2023): "Customized influence maximization in attributed social networks: heuristic and meta-heuristic algorithms", in: Complex & Intelligent Systems, vol.10, No. 10.

[4] Cheng, Juan (2023): "Dynamic Path Optimization Based on Improved Ant Colony Algorithm", in: Journal of Advanced Transportation, pp. 1-11.

[5] Dorigo, Marco & Stützle, Thomas (2010): "Ant Colony Optimization: Overview and Recent Advances", in: Handbook of Metaheuristics, pp. 227-263.

[6] Cheng, Juan (2023): "Dynamic Path Optimization Based on Improved Ant Colony Algorithm", in: Journal of Advanced Transportation. pp. 1-11.

[7] Karpov, Kirill & Kachan, Dmitry & Siemens, Eduard (2023): "Enhancing Application Layer Multicast Using Multi-objective Optimization Techniques", in book: Information and Communication Technologies and Sustainable Development, pp. 42-53.

[8] Romanov, Oleksandr & Saychenko, Ivan & Marinov, Anton & Skolets, Serhii (2021): "Research of SDN network performance parameters using Mininet network emulator", in: Information and Telecommunication Sciences, pp. 24-32.

[9] Rowshanrad, Shiva & Abdi, Vajihe & Keshtgari, Manijeh (2016): "Performance evaluation of SDN controllers: Floodlight and OpenDaylight", in: IIUM Engineering Journal, vol. 17, pp. 47-57.

[10] Abdullah, Taher (2014): "Testing of Floodlight controller with Mininet in SDN topology", in: ScienceRise, vol. 5, pp. 68-73.

[11] Hojjat, Hossein & Nakhost, Hootan & Sirjani, Marjan (2006): "Formal Verification of the IEEE 802.1D Spanning Tree Protocol Using Extended Rebeca", in: Electronic Notes in Theoretical Computer Science, vol. 159, pp. 139-154.