Evaluating Trade-offs for Green Routing in Communication Networks

Jan Kitanovski*, Kaspar Zimmermann*, Line M. P. Larsen† and Sarah Ruepp*
* Department of Electrical and Photonics Engineering, Technical University of Denmark, Kgs. Lyngby, Denmark
† Department of Mobile Innovation, TDC NET, Copenhagen, Denmark
e-mail: s232649@dtu.dk, s232759@dtu.dk, lil@tdcnet.dk, srru@dtu.dk

***Abstract*—With the demand for high-speed, high-capacity networking increasing each year, it is important to focus on the infrastructure and software running on the networking backbone. A crucial component of this focus is the need for efficient pathfinding algorithms, which can determine the best route for data to travel across a network, and ensuring optimal resource usage and performance. This paper promotes the importance of selecting the best routing algorithm for a specific purpose of data transport, highlighting the trade-offs involved in green routing. It also presents the work and results of a network simulator that uses four different algorithms (Dijkstra's, A*, Floyd-Warshall, and Depth-First Search) to determine which performs best in a set environment. In terms of latency, Floyd-Warshall showed a 64% improvement over Dijkstras, whereas A* showed a 57% improvement over Dijkstra's. Results indicate that the tradeoff of choosing an algorithm with a lower latency can also result in higher carbon cost.**

***Keywords*—Green-Routing; Denmark; Path-Finding; algorithms; latency; overhead; carbon cost.**

I. Introduction

As the number of Internet-connected devices continues to rise globally, the data throughput generated by these devices has significantly increased. In 2022, it reached nearly 1200 Tbit/s globally [1]. Consequently, new network infrastructure needs to be built, which also consumes energy. Data centers and other similar technologies used 460TWh in 2022, which accounted for almost 2% globally [2]. Therefore, it is important to focus on enabling greener alternatives to how the networks operate, as this might also impact on the end-cost of operating the infrastructure. Choosing green routing-themed research comes from the need to address the impact of the growing network infrastructure. With a big shift towards renewable energy sources, it is important to also optimise the network performance itself.

Routing is a way of finding the path from one node or end device, through the network to the end-point of the data. Green-routing improves on this, by also taking into account where the energy, powering the devices that enable this transmission comes from, and how it can be used in the most efficient way.

In this project, an event-based simulator will be programmed to simulate the network traffic within Denmark. The data used for the simulator will be based on the data centers and other smaller nodes spread throughout the country. Due to data gathering limitations, different alternative approaches had to be used to simulate the network behaviour.

*The first two authors have contributed equally to this paper

This paper describes a simulation project that focuses on green routing algorithms and compares them in terms of performance and efficiency.

The simulator operates as an event-based simulation, designed to manage large-scale simulations. Each event is assigned specific attributes before being queued for simulation. These attributes include the time of occurrence, which determines when the event will be processed, and the type of event. Event types include original packet creation, which initiates the start of the algorithm search. Normal packet creation can generate either an overhead packet or a data packet, depending on the algorithm's progress. Data packets can be generated with either high or low priority. High-priority packets are routed through the shortest path to minimize latency, while low-priority packets are directed through the greenest route to minimize environmental impact.

For the remainder of this paper, Section II provides the state of art, Section III dives into the methodology used in the event-based simulator, Section IV provides the Testbed setup of the simulation, Section V overlooks the results gathered from the simulation, which are then discussed in Section VI and concluded in Section VII.

II. State of the Art

Several studies have explored various aspects of the green routing problem, each approaching it differently. Zhu et al. [3] examined the development of an energy-aware network management platform, OpenNaaS, which supports SDN (Software Defined Networking) to create green-greedy routing paths. Their system measures energy, cost, and sustainability information for networks, demonstrating the platform's potential for energy-efficient routing in large-scale networks. Wang et al. [4] focused on power saving and QoS (Quality of Service) for many-to-many multicast in backbone networks. They developed the GIQM (Green Intelligent flexible QoS many-to-many Multicast routing algorithm), which uses power consumption as a routing metric and supports flexible QoS requirements. Their algorithm outperformed other schemes such as the CBT (Core-Based Trees algorithm) in power savings and routing success. In their research, Yang et al. [5] devised hop-by-hop algorithms to achieve loop-free routing, minimizing energy usage. They were innovated upon the Dijkstra's algorithm, by creating various "Dijkstra-green" versions to improve the routing efficiency. Lee et al. [6] proposed the DEAR algorithm, which improves energy efficiency while meeting flow delay requirements in networks with diverse energy profiles.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

The DEAR algorithm effectively identifies the least energy-consuming path while ensuring flow delay requirements are satisfied. Hossain et al. [7] proposed a sustainable method for greening the Internet by introducing "pollution-aware routing," which integrates considerations of carbon emissions and non-renewable energy usage into traditional energy-aware routing. Their holistic approach, implemented with SDN, demonstrated significant reductions in CO2 emissions compared to conventional energy-aware solutions. Nwachukwu et al. [8] has focused on the optimization for simultaneous routing and bandwidth allocation through the use of Lagrangian methods. Their work shows that augmented Lagrangian algorithms are highly effective on this matter.

Upon reviewing existing research in this field, it was found that the primary focus has been on developing new technologies, algorithms, and enhancements to these algorithms, with relatively few studies dedicated to comparing the various path-finding algorithms.

The objective of this research paper is to develop an event-based simulator capable of incorporating various data, such as the locations of data center nodes within Denmark, and simulating network behaviour using different algorithms. The aim is then to compare the effectiveness of each algorithm based on multiple result variables, including total overhead packets, carbon cost per data transmitted, carbon cost distribution between data and overhead and latency.

III. Methodology

A. Green Routing

Green routing as a principle focuses on the ability of the network to choose transportation routes that minimize the environmental impact, by reducing fuel consumption and emissions. It utilizes different path-finding/optimization algorithms as well as data of the types of power sources available for powering the network. In this paper, the evaluation of these algorithms will be based on multiple values, such as the latencies, total overhead packets, total data packets, carbon emission of the transmission, expressed as the carbon cost. These values of course will be expressed as multiples of minimum, maximum and average values. These values will be represented in the corresponding figures.

B. Algorithms

In this research, the main focus will be on the comparison of results, efficiencies, and drawbacks of various path-finding algorithms. The algorithms in question will be: Dijkstra's, Floyd-Warshall, an advanced version of Dijkstra's, A*, and the DFS (Depth-First Search) approach. Figure 1 shows an example graph that the operation of the algorithms will be explained on.

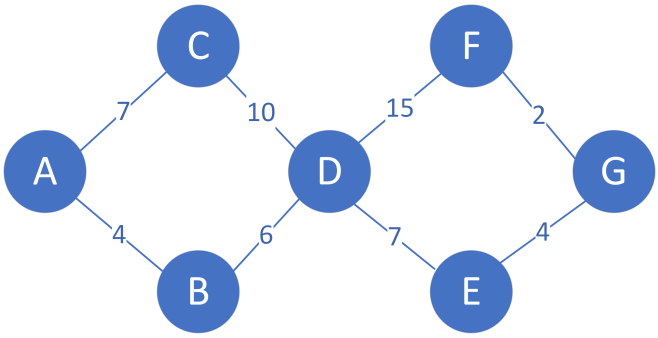

Figure 1. An example graph with 7 nodes and positive edges, used to illustrate algorithm operation

- Dijkstra's finds the shortest path starting from a single point and continuing by the use of the smallest known distance to other nodes in a weighted graph. When a shorter route to a node is found, the table is updated, until all paths to the end node are found. Starting from node A, the algorithm will choose the shortest distance, which is to node B. The next shortest step is to node D, after which the algorithm sets the shortest distance to D as 10. The algorithm keeps taking the shortest path available to it and when it gets to a node that is already discovered, it will update the distance to it, if it is shorter. For example, when the path from C to D is discovered, its distance of 17 is compared to the previously found distance of 10, and no changes are made [9]. Dijkstra's is tested because it is a very commonly used shortest-path algorithm and is easy to implement.
- A* extends Dijkstra's algorithm by incorporating heuristics, which are used to guide the search towards the goal, and utilizes tables to keep track of visited and non-visited nodes. When A* finds the path in the Dijkstra's searching method, it will save it. For example, it will know that the path from A to D is shortest through B, so next time it won't search the entire graph again and will use that path [10]. A* is used to see if it is possible to fix some of the shortcomings of Dijkstra's, but not lose the accuracy.
- The Floyd-Warshall algorithm calculates the shortest paths between all pairs of nodes in a weighted graph. It initializes the distance matrix with the weights of direct connections between nodes, using infinity for pairs of nodes without a direct connection and zero for the distance from a node to itself. The algorithm then updates this matrix by considering each node as an intermediate point and checking if a path through this node offers a shorter route between any two nodes. For example in the given graph, the algorithm starts by setting the initial distances, such as A to C to 7, A to B to 4, and A to D to infinity. It then updates paths A to D by taking into account intermediate nodes. For example, it might find that the path from A to D through C (with a combined weight of 17) or through B (with a combined weight of 10) is shorter than the initially set distance. In this case, the path A to B to D has a total weight of 10, which would be the updated shortest distance from A to D [11]. Floyd-Warshalls algorithm was chosen, because it is a competing shortest-path algorithm to Dijkstra's. Their difference should mostly depend on the density of

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

the graph, so results may change depending on the chosen network size.

- DFS initiates by choosing a point of entry and randomly continuing node by node in one direction. If a connection is not successfully established, it backtracks its hops and tries again. Starting from node A, it will randomly choose node C or B. Then, it will keep going with random choices, without backtracking. If it finds the end node, it will use whatever path it took even if its not the most efficient one. DFS was chosen as a reference point to use for comparing the other algorithms. It shows why, in general, using a routing algorithm is better than randomly choosing which way to go.

IV. Testbed Setup

Python was used to set up the simulation. Additional libraries, such as heapq and pandas were used for efficient priority queue operations and data manipulation and analysis, respectively, while Matplotlib was used to visualise the results.

The data for the simulator was gathered from various sources, such as different research papers and publicly available data, which are explained below. However, very precise data was not available, so reliance on other models for electrical energy sourcing, including the estimation of the ”greeness of sources,” and the connection between the bandwidth created and population densities in certain areas was necessary.

The carbon cost calculation was done with the use of the nearest energy sources to the node and multiplied by the total distance travelled. The data has been gathered by the Danish energy agency [12].

The latency calculation was done by combining the data for the propagation of light through optic fiber of 500 microseconds per 100km [13] and the average latency introduced by the processing time of a switch set to 2 microseconds [14].

It is worth noting that the connections between datacenters in Denmark were established based on a publicly available map [15], which was transcribed into the setup shown in Figure 2. For the simulator itself, a few different configurations were chosen, however the main difference between them was the number of simulations to run.

The estimation for electrical energy sourcing was derived from a model using the closest sources of electrical power. In this approach, a specific area around the data center, with an estimated set power consumption, was examined. A percentage-wise distribution of power sources, which could be from renewables or non-renewables, was utilized. A similar approach was used for estimating the amount of data generated at a certain node in the network. This estimation was based on the map of Denmark, with population densities being utilized for the calculations.

The tests are run for 10,000 seconds to minimize the effect of the semi-random packet generation without losing the realistic traffic. All tests were run at least 5 times to check for anomalies. This was normally enough, as the long simulation time removed any randomness from the results.

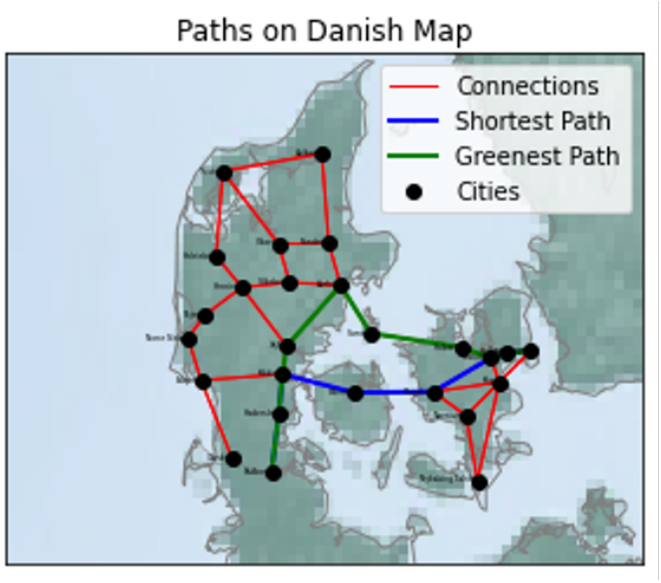

Figure 2. The main network nodes of Denmark, showing the use of Dijkstra's algorithm for path-finding the lowest latency route, as well as the greenest path.

The graph used for the algorithm uses the map of databases and also places nodes in locations where fibre optic cables meet at larger settlements. It consists of 26 points with each having 1 to 5 connections to nearby nodes.

V. Results

For the results, three main metrics were considered: Overhead packets, which show non-data packets used by each algorithm, Carbon cost, to show how ”Green” the algorithm is and Latency to show how fast the algorithm is.

1) Overhead packets: The test results for total overhead packets created by the algorithms in the simulation time are shown in Figure 3. The amount of overhead packets created can indicate how much additional traffic the algorithm creates for the entire network.

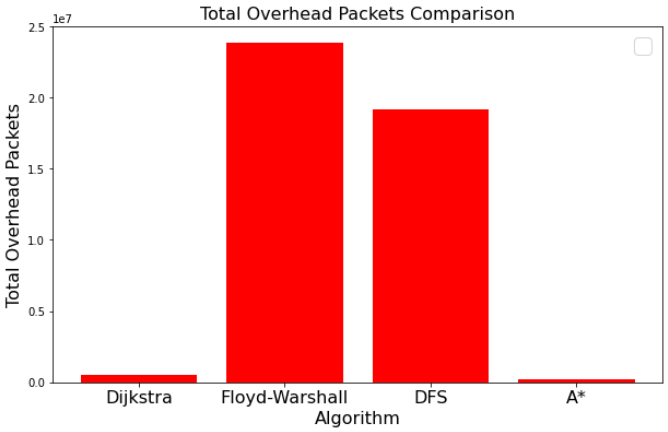

Figure 3. Total overhead packets sent by each algorithm during the duration of the simulation

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

Floyd-Warshall produces the highest amount of overhead because it uses overhead packets to find the cost between every pair of nodes. The random design of the DFS approach creates around 23% less overhead than Floyd-Warshall. Both of these approaches created significantly more overhead than the Dijkstra-based approaches and would require a lot more processing by all nodes in the network.

Dijkstra's algorithm creates 97.86% less overhead than Floyd-Warshall and therefore causes less congestion of the network. The implementation of A* creates 53% less overhead than Dijkstra's and 99% less overhead than Floyd-Warsall, which is expected, as even though it searches all connections in the network, it only does it every 250 seconds. Using the saved routes later means that no extra overhead is created and the capacity can instead be used for data. This might cause problems if a larger part of the network is disconnected right after the full search, which is why the precise time of searches should be balanced according to the network requirements.

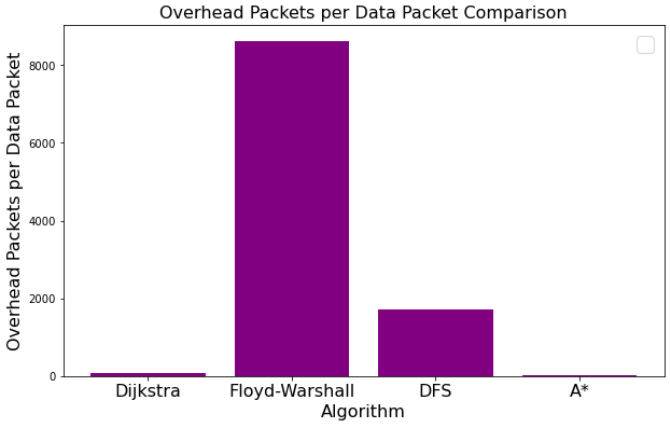

Figure 4. Total Carbon cost per data packet sent by each algorithm during the simulation

Figure 4 illustrates that DFS sends 80% fewer overhead packets per data than Floyd-Warshall. This is due to the non-optimal paths that the data chooses in the DFS algorithm, which means the data travels more in the graph.

2) Carbon costs: The results from the carbon cost calculations can be seen in Figure 5. As overhead packets carry less information than data packets, the carbon cost of overhead is set to 1/1000 of data.

The largest carbon cost is produced by the DFS algorithm. It uses a lot of carbon for overhead packets used for searching, and as the routes it finds are not the most optimal, it also uses the most on data. Although Floyd-Warshall uses the least amount of carbon on data, its large amount of searching means that it has the second highest carbon cost, which is still 99% less than DFS. It is evident though that if the overhead packets were even smaller compared to data, then Floyd-Warshall could have the smallest carbon cost. In this simulation, A* generates 40% less overhead than Floyd-Warshall. Since the graph of nodes is quite small, the performance of A* is worse than Dijkstra's by 38% as the benefits of the heuristic model

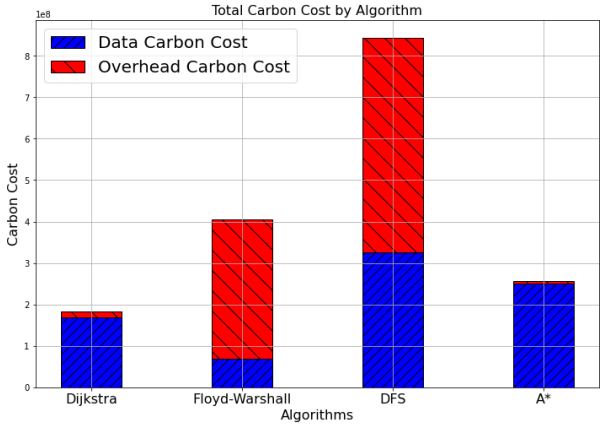

Figure 5. Carbon cost distribution showing each algorithms carbon cost from data and overhead packets

do not appear. The carbon cost of Dijkstra's is the smallest, although the paths it finds are not the most optimal, it uses less overhead than Floyd-Warhsall, which makes it overall 57% more "Green".

3) Latency: The third parameter that shows the efficiency of the algorithm is latency. Latency in this case takes into account the search time and the data travel time. Simulation results are shown with maximum and minimum limits in Figure 6.

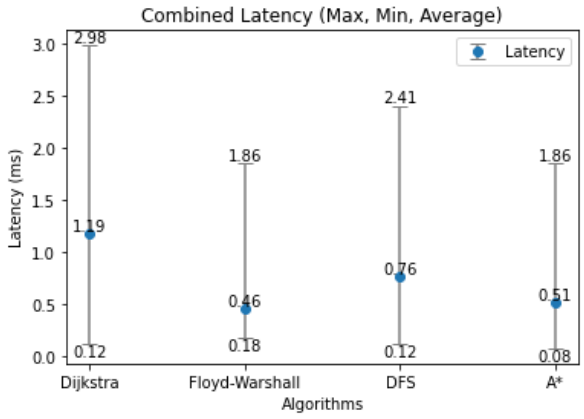

Figure 6. Latency results with average latencies and maximum and minimum latencies shown as limits

The largest average and maximum latency is by Dijkstra's algorithm, which is 55% more than the second-best DFS. The search starts from one point and systematically searches through the entire network. Floyd-Warshall has the lowest average latency due to how the algorithm searches, with a 64% improvement on Dijkstra's. As it starts from several nodes at the same time, it is faster than Dijkstra's. However, as it still needs to search most of the graph, it has the highest minimum latency. DFS has the second-highest average and maximum latency. This is due to the randomness, which worst case will search the entire graph, but on average will only search half of it. A* has the lowest minimum latency due to using pre-

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org

saved paths, which do not require searching. The case where A* searches the entire network increases the average latency, but it is still on average 57% faster than Dijkstra's.

VI. Discussion

These results highlight that with each algorithm there are positives and negatives. If the impact of overhead packets is negligible and their size is small, then only the efficiency of the routing is important. In this case, Floyd-Warshall could be used for extensive searching. In case the goal is to minimize the amount of overhead, using a version of A* or Dijkstra's is preferable. However, when choosing Dijkstra's the network will have longer latency, which should not be used for real-time communication. A* can solve that problem with the use of saved paths and heuristics. To make A* more efficient, a larger network is needed, where the model can improve on Dijkstra's. The tradeoffs of each algorithm are illustrated in Figure 7. With the X-axis showing increasing total carbon emissions and the Y-axis showing the average latency, it is best if the algorithms aren't in the extremes in any axis. Dijkstra's is on the top left of the graph, which means that although the total carbon emissions are low, the latency is significantly worse than the others. Similarly, DFS is on the extreme of carbon emissions. Floyd-Warshall and A* both do rather well, but there is some noticeable difference in carbon emissions. In general, as A* has a suitable average latency and low carbon cost, it can be preferred over the other algorithms tested.

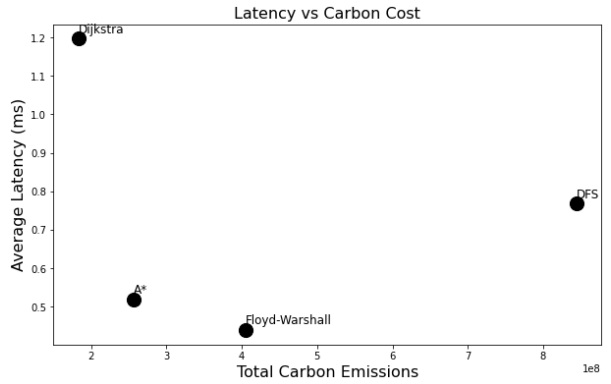

Figure 7. Carbon cost and latency comparison for all algorithms with carbon cost on the x-axis and latencies on the y-axis

VII. Conclusion

This paper was set to address the environmental impact of the expanding ICT (Information and Communication Technology) infrastructure by comparing different path-finding algorithms.

The findings indicate that the Floyd-Warshall algorithm produces the most overhead packets, followed by Depth-First Search, with a decrease of 23%, while A* has the fewest, with 99% less overhead packets than Floyd-Warshall. In terms of carbon cost, the DFS algorithm generates the highest total carbon cost, and A* having a 38% worse performance than Dijkstra's. Interestingly, Floyd-Warshall has a lower carbon cost for data packets than Dijkstra's; however, the higher number of overhead packets increases its total carbon cost. Regarding latency, Floyd-Warshall has the lowest average latency with an improvement of 64% over Dijkstra's, while Dijkstra's shows the largest range of values. A* also shows a 57% improvement over Dijkstra's, even in the situation, where it searches the whole network.

These results demonstrate that no single algorithm is universally optimal, as each one has its strengths and weaknesses. However, as A* has good latency and carbon cost metrics, it can be the preferred algorithm for most cases.

Acknowledgment

Support from Innovation fund Denmark, through grant no. 1045-00047B, as well as the "Innovative solutions for next generation of Green COMmunications infrastructures" GREENCOM project is gratefully acknowledged.

References

[1] ITU, "Unrelenting global consumption of Internet data continues to drive demand for international bandwidth usage," https://www.itu.int/itu-d/reports/statistics/2022/11/24/ff22-international-bandwidth-usage/ [retrieved: June, 2024], 2022.

[2] IEA, "Electricity - Analysis and forecast to 2026," https://iea.blob.core.windows.net/assets/6b2fd954-2017-408e-bf08-952fdd62118a/Electricity2024-Analysisandforecastto2026.pdf[retrieved: June, 2024], 2024.

[3] H. Zhu, J. Aznar, C. de Laat, and P. Grosso, "Green routing in software-defined data center networks based on opennaas," *Software Networks*, 2015.

[4] X. Wang, J. Zhang, M. Huang, and S. Yang, "A green intelligent routing algorithm supporting flexible qos for many-to-many multicast," *Computer Networks*, vol. 126, pp. 229–245, 2017.

[5] Y. Yang, M. Xu, D. Wang, and S. Li, "A hop-by-hop routing mechanism for green internet," *Ieee Transactions on Parallel and Distributed Systems*, vol. 27, no. 1, pp. 2–16, 2016.

[6] E. J. Lee, Y. M. Kim, and H. S. Park, "Dear: Delay-guaranteed energy profile-aware routing toward the green internet," *Ieee Communications Letters*, vol. 18, no. 11, pp. 1943–1946, 2014.

[7] M. M. Hossain, J. P. Georges, E. Rondeau, and T. Divoux, "Energy, carbon and renewable energy: Candidate metrics for green-aware routing?" *Sensors (switzerland)*, vol. 19, no. 13, p. 2901, 2019.

[8] A. C. Nwachukwu and A. Karbowski, "Solution of the simultaneous routing and bandwidth allocation problem in energy-aware networks using augmented lagrangian-based algorithms and decomposition," *Energies*, vol. 17, no. 5, p. 1233, 2024.

[9] GeeksforGeeks, Sanchhaya Education Private Limited, "What is dijkstra's algorithm? — introduction to dijkstra's shortest path algorithm," https://www.geeksforgeeks.org/introduction-to-dijkstras-shortest-path-algorithm/ [retrieved: June, 2024], 2024.

[10] ——, "A* search algorithm," https://www.geeksforgeeks.org/a-search-algorithm/ [retrieved: June, 2024], 2024.

[11] ——, "Floyd warshall algorithm," https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/floyd-warshall-algorithm [retrieved: June, 2024], 2024.

[12] The Danish Energy Agency, "Power production and transmission in denmark," https://ens.dk/en/our-services/statistics-data-key-figures-and-energy-maps/energy-infomaps [retrieved: June, 2024], 2019.

[13] F. Azendorf, A. Dochhan, and M. H. Eiselt, "Accurate single-ended measurement of propagation delay in fiber using correlation optical time domain reflectometry," *Journal of Lightwave Technology*, vol. 39, no. 18, pp. 5744–5752, 2021.

[14] T. Hegr, M. Voznak, M. Kozak, and L. Bohac, "Measurement of switching latency in high data rate ethernet networks," *Elektronika Ir Elektrotechnika*, vol. 21, no. 3, pp. 73–78, 2015.

[15] DataCenterJournal, "Map of Denmark's Data Centers," https://www.datacenterjournal.com/data-centers/denmark/ [retrieved: June, 2024], 2024.

Courtesy of IARIA Board and IARIA Press. Original source: ThinkMind Digital Library https://www.thinkmind.org