

Active GA accelerated by Simulated Annealing to solve SPP in packet networks

Daniel S. Fonseca¹, Elizabeth F. Wanner¹, Carolina G. Marcelino^{2,3},
Gabriel P. Silva², Silvia Jimenez-Fernandez³, and Sancho Salcedo-Sanz³

¹ Federal Center of Technological Education of Minas Gerais (CEFET-MG),
Department of Computing, Brazil. efwanner@decom.cefetmg.br

² Federal University of Rio de Janeiro (UFRJ), Institute of Computing, Brazil
{[carolina](mailto:carolina@ic.ufrj.br),[gabriel](mailto:gabriel@ic.ufrj.br)}@ic.ufrj.br

³ University of Alcalá (UAH), Dep. of Signal Processing and Communication, Spain
{[silvia.jimenez](mailto:silvia.jimenez@uah.es),[sancho.salcedo](mailto:sancho.salcedo@uah.es)}@uah.es

Abstract. This paper presents two approaches to deal with the shortest path problem (SPP) solution for routing network packets in an optimized way. The first one uses Simulated Annealing (SA), and the second one is a novel hybridization of the Genetic Algorithm with Dijkstra mutation accelerated by the SA (SGA). Also, two different case scenario configurations, each with 144 nodes, are employed to assess these two proposals, and the total time spent to fill out the routing tables, referring to the transmission of a packet from the initial to the destiny nodes, is measured. A statistical comparison is applied to identify differences among the algorithm's solutions. Experiments and simulations have shown that the SGA presented competitive results compared to standard SA and can solve the problem with fast convergence, which makes us conclude that it can operate efficiently in actual computer networks.

Keywords: Optimization · evolutionary algorithms · shortest path problem.

1 Introduction

On Internet, routing is the process by which computers receive and deliver packets of information. The Internet uses a routing model named *hop-by-hop*. In this model, each computer or router examines the packet's destination address, executes a routing algorithm to determine the next hop, and delivers it to the next router or computer, where the process will be again performed. Efficient routing algorithms can use diverse techniques to find an adequate route path for packet transmission [1].

Routing tables and protocols are necessary to achieve the expected results of the transmission process. Network members' addresses are the most relevant information in the routing tables. Besides the contents of the routing table, a protocol also defines the best routes for delivering packets. As an assessment of the quality of service, some performance metrics inherent to the network's

communication, such as the number of hops, delays, and transmission costs of a given packet, are also considered [2]-[3].

1.1 Related works

The use of software agents for network routing produces better results than conventional approaches, as shown by many studies [4] [5]. These algorithms have proven to be viable candidates for solving problems of dynamic and unpredictable nature, such as routing in large networks. It is mainly due to the continuous active search for non-local information that agents allow. Network routing is the mechanism chosen to send packets from any source to a destination in the network. Lopez and Heisterkamp [6] proposed a solution using the Simulated Annealing (SA) algorithm with hierarchical Q-routing, producing a dynamic routing protocol. In a 6x6 grid topology, this approach efficiently provided the route tables in a short time.

Flying Ad-hoc Networks are a special type of wireless network that, in general, has been using optimization algorithms to establish efficient and dynamic routes. Following Q-routing ideas, Rovira-Sugranes et al. [7] concluded that SA adapts to the network dynamicity without requiring manual re-initialization at transition points. SA showed to be an alternative routing algorithm in wireless sensor networks lifetime [8] and flying-software defined networks [9]. Raj and Rahimunnisa [10] proposed a hybrid approach coupling the Genetic Algorithm (GA) and SA to solve the routing packages in wireless Ad-hoc networks. Sundar and Kathirvel [11] proposed a delivered mechanism over variable length density using the SA in mobile ad-hoc network. The efficiency of coupled GA and SA was verified by Prasad and Rayanki [12] when applied to computer network routing.

Hamed [13] proposed a genetic version of an algorithm to find the k-shortest paths in a network. Zhang et al. [14] applied GA in a small simulation to improve the quality of service in computer networks, reporting effective results. A multicast routing algorithm, also using genetic algorithms, was proposed using an approach with both bandwidth and delay constraints [15]. Recently, some works used GA with many goals: to minimize the construction of network routing tables [16], to provide an energy-efficiency network protocol [17], and to improve the efficiency in routing packets [18]-[19]. GA was successfully used for routing Internet of Things (IoT) networks [20]. A new network protocol based on GA was proposed in [21]. Results showed that the time to construct network routes was reduced in 12.74%. Energy consumption should be considered as one of the foremost vital limitations in autonomous systems, and to provide a solution for that Bhardwaj and El-Ocla [22] proposed a multipath routing protocol based on GA.

This work objective is to solve the shortest path in computer networks. To achieve this goal, we propose a genetic algorithm with a new mutation scheme inspired by Dijkstra's algorithm [23] – the Genetic Algorithm with Dijkstra mutation (GA). To accelerate the search space, we couple GA with an SA mechanism. To the best of our knowledge, this is the first time that a mixed approach between GA and Dijkstra algorithms is reported. To validate our proposal, we

conduct an experimental simulation comparing the results obtained by the three evaluated algorithms. Specifically, in our simulation, we adopt an experimental scenario, using a topology with 144 interconnected routers, differentiating the approach taken in other related works, which used a maximum of 30 routers. In our perspective, the size of the 144-node network is well-suited for the simulated experiments.

This work is organized as follows: Section 2 describes the shortest path problem and presents the optimization modeling proposed. Section 3 describes the standard Simulated Annealing and addresses the coupled approaches. Section 4 explains the simulation methodology adopted. The experiments carried out and the data analysis are presented in Section 5. Finally, Section 6 concludes the report and presents future perspectives.

2 Problem Description

The shortest path problem can be described as finding the shortest path that satisfies a given condition. In this work, we propose a mixed meta-heuristic to solve this problem, always keeping in mind the real characteristics of the Internet. This heuristic is based on the analogy of finding the shortest path between two nodes in a computer network. So, meta-heuristics is an emerging idea able to solve this type of problem. This approach differs from the algorithms reported in the internet protocols literature.

Usually, conventional methods compare all the possibilities to find the best solution. This mechanism is time-consuming, increasing the delay for a network with a large number of nodes and edges. In this context, computer networks can be described as a weighted directed graph $G = (N, E)$, in which N indicates the set of nodes and E indicates the set of links connecting the nodes. $|N|$ and $|E|$ denote the number of nodes and links in the network, respectively [24].

2.1 Shortest path Optimization Modeling

In this work we propose an optimization model to solve the shortest path problem, having the previous graph notation to computer networks in mind. In graph theory, the minimal path problem is characterized by finding a path between two nodes such that the sum of weights of its edges is minimum. In this way, an objective function of the optimization problem can be defined by Eq. (1),

$$dist(v, i) = \min_{(u,v) \in E} \{dist(u, i-1) + \ell(u, v)\} \quad (1)$$

in which, for each vertex v and each integer $i \leq k$, $dist(v, i)$ is the shortest path from source vertex s to v using i edges, $dist(u, i-1)$ is the shortest path from source vertex s to u using $i-1$ edges, and $\ell(u, v)$ is the distance between u and v .

In our optimization model, all the paths are valid as their bandwidth availability is checked before finding out their delay factor,

$$AveragePacketDelay = \frac{Delay}{number\ of\ links} \quad (2)$$

in which,

$$Delay = \sum_{i=1}^n \frac{BandwidthAvailable_i}{DataSize}.$$

Fitness is calculated through Total Cost Time, in seconds (s):

$$Fit = \min \sum_{i=1}^n (AveragePacketDelay + TimeRoute). \quad (3)$$

The optimal path selection is mainly based on the AveragePacketDelay term and in the TimeRoute spent at each link where the route walks. If there are two or more paths with the same fitness value, the algorithm will choose the one with fewer hops.

3 Meta-heuristics addressed

A meta-heuristic can be defined as a high-level problem-independent set of guidelines and strategies to design heuristic optimization algorithms [25]. Meta-heuristics can combine several mechanisms as diversity, memory, multiple populations, and self-adaptive schemes [26]. This work addresses two distinct meta-heuristics: a proposed Genetic Algorithm with Dijkstra mutation (GA), and the standard Simulated Annealing (SA) [27] to solve the shortest path problem in computer networks with comparative performance analysis.

3.1 The proposed Genetic Algorithm with Dijkstra mutation

Genetic algorithms (GA) are search meta-heuristics utilized to find approximate solutions to general optimization problems. GA is based on evolutionary biology with genetic heritage, mutation, natural selection, and breeding [28]. GA is implemented as a computer simulation in which a given population of abstract representations of solutions is selected in the search for better solutions.

The evolution process usually starts with a randomly generated set of solutions and occurs throughout generations. As generation passes by, the fitness of each solution in that generation is evaluated. Generally, the operators in GA are crossover, mutation, selection, and elitism. The new population is used as an input to the next iteration of the algorithm. The evolutionary loop is executed until one or more solutions meet the expected result by the implemented objective function. Algorithm 1 presents a pseudo-code for the Genetic Algorithm.

To solve the specific minimal path problem, in which the individual is a route vector that contains nodes, the mutation and crossover operators are explained as follows. In our GA implementation, the mutation operator has a Dijkstra

Algorithm 1: Genetic algorithm pseudo-code

```

1 begin
2   Initialize current population;
3   Evaluate the fitness of each individual;
4   Store the individual with highest fitness;
5   while Stop criteria is not met do
6     for 0 to population size do
7       Select to individuals to breed;
8       Apply crossover to the individuals;
9       Apply mutation to the children if necessary;
10      Save the children for the next generation;
11      Evaluated children fitness;
12    end
13    Current generation replaced by next generation;
14    Evaluate mean fitness;
15    if Any fitness in the population > best fitness then
16      Erase previously stored best individual;
17      Store new best individual;
18    end
19  end
20 end

```

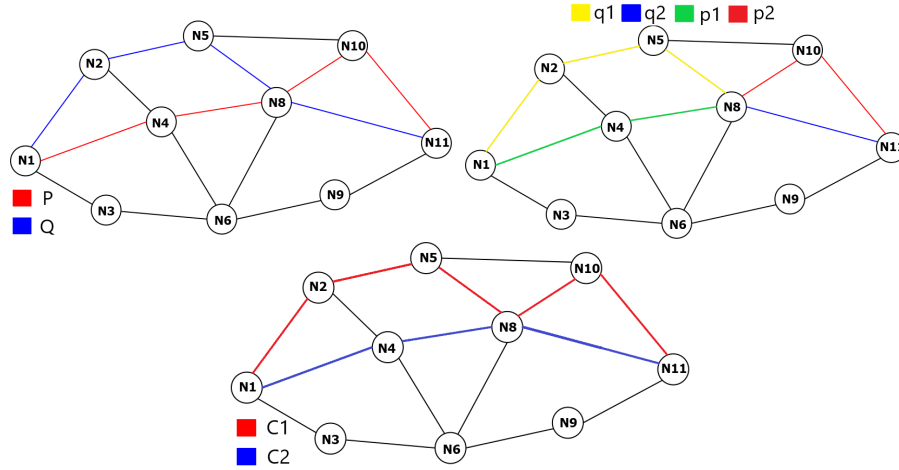


Fig. 1: Let $P[N1-N4-N8-N10-N11]$ and $Q[N1-N2-N5-N8-N11]$. Choice a partition point to cut P and Q in $p1[N1-N4-N8]$, $p2[N8-N10-N11]$, $q1[N1-N2-N5-N8]$, $q2[N8-N11]$. The new offspring $C1[N1-N2-N5-N8-N10-N11]$ and $C2[N1-N4-N8-N10-N11]$ will be generated.

structure to provide an improvement in the GA cycle. We are proposing a merged approach that uses the best characteristics of both algorithms. Fig. 1 and 2 show the crossover and mutation operators, respectively.

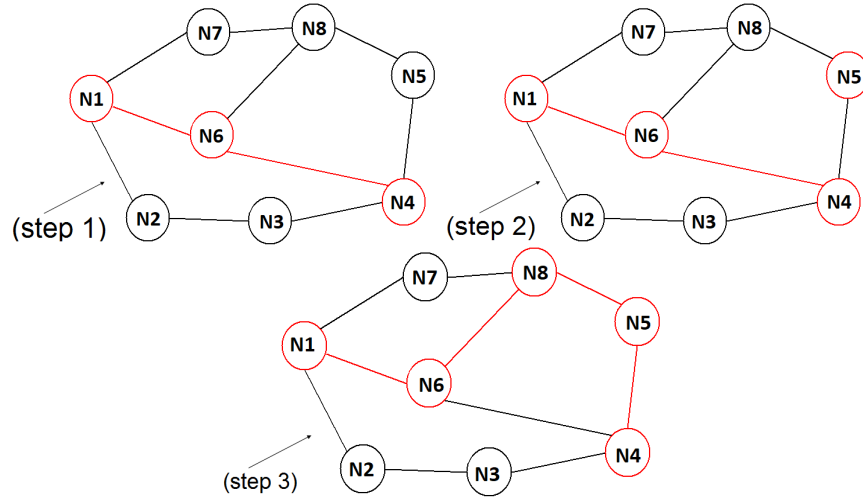


Fig. 2: Given the route N1-N6-N4, in the mutation process, its randomized that the node N5 will be added between N6 and N4, thus the route from N6 to N5 (by N8) and N5 to N4 must be calculated to construct a valid route.

CROSSOVER:

1. Given two routes parents P and Q with a common node;
2. Separate the route P in p1 and p2, from the start to the common node and from the common node to the destination respectively;
3. Separate the route Q in q1 and q2, from the start to the common node and from the common node to the destination respectively;
4. Generate 1st offspring (C1) with combination of p1 and q2;
5. Generate 2nd offspring (C2) with combination of q1 and p2.

MUTATION:

1. Select a random position in the route;
2. Insert a random node in this position;
3. Calculate the route from the previous node to the new inserted node using Dijkstra [23] and insert it in the route.

3.2 Standard Simulated Annealing

Simulated annealing is a well-studied local search meta-heuristic applied to a large variety of optimization problems. The key feature of this algorithm is that it provides a mechanism to escape local optima by allowing hill-climbing moves that worsen the objective function value in search of global optima.

The name comes from an analogy to the process of physical annealing with solids, in which a crystalline solid is heated and then allowed to cool very slowly until it achieves its most regular possible crystal lattice configuration and thus is

free of crystal defects. Simulated annealing establishes the connection between this type of thermodynamic behavior and the search for global minima for a discrete optimization problem.

At each iteration of a simulated annealing algorithm, the values for two solutions (the current solution and a newly selected solution) are compared. Improving solutions are always accepted, while a fraction of non-improving solutions are accepted in the hope of escaping local optima in search of global optima. The probability of accepting non-improving solutions depends on a temperature parameter, which is reduced with each iteration of the algorithm [27]. Algorithm 2 presents a pseudo-code for the Simulated Annealing.

Algorithm 2: Simulated annealing pseudo-code

```

1 begin
2   Select an initial solution  $\omega$ ;
3   Select an initial temperature  $T$ ;
4   Select the temperature reduction factor,  $\alpha$ ;
5   Select a repetition schedule,  $M_k$ ;
6   Select a maximum number of iterations,  $max_{it}$ ;
7    $k = 0$ ;
8    $t = T$ ;
9   while  $k < max_{it}$  do
10    for  $m = 0$  to  $M_k$  do
11      Generate a solution  $\omega'$ ;
12      Calculate  $\Delta_{\omega, \omega'} = f(\omega') - f(\omega)$ ;
13      if  $\Delta_{\omega, \omega'} \leq 0$  then
14         $\omega = \omega'$ ;
15      end
16      if  $\Delta_{\omega, \omega'} > 0$  then
17         $\omega = \omega'$  with probability  $exp(-\Delta_{\omega, \omega'}/t)$ ;
18      end
19    end
20     $t = \alpha \times t$ ;
21     $k = k + 1$ ;
22  end
23 end

```

In the present work, to generate a new solution ω' , the mutation operator from the genetic algorithm is applied in the previous solution ω .

3.3 Active Genetic Algorithm accelerated by Simulated Annealing

In this work, we also propose a method to combine the advantages of each of the previously presented algorithms. The GA should take longer times to run due to its populational nature, while the SA should be able to find a local optimum faster.

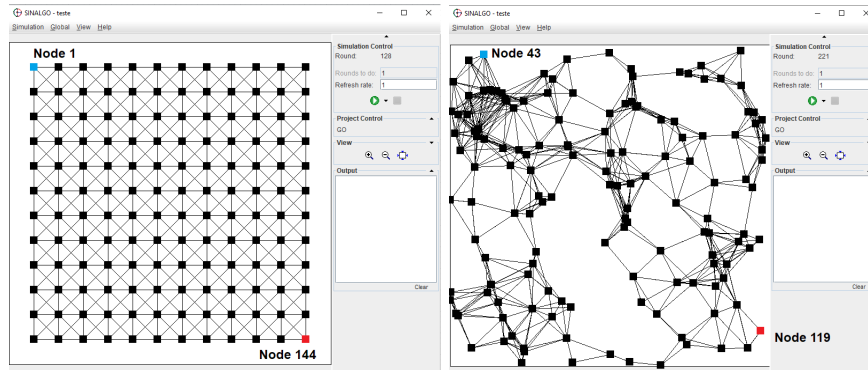
Given this hypothesis, we propose a method to use the results obtained quickly in the first iterations of the simulated annealing as the initial population for the genetic algorithm, allowing it to converge faster. We call this method Active Genetic Algorithm Accelerated by Simulated Annealing – SGA.

In this method, the simulated annealing is applied to a portion of the individuals in the initial population of the genetic algorithm, thus allowing the genetic algorithm to start with improved solutions probably closer to the optimal solution. The SGA also implements a new stop criterion: the algorithm terminates when all individuals converge to the same result, thus allowing the algorithm to execute fewer generations.

4 Methodology and Simulation

The methodology used to validate the efficiency of each meta-heuristic is based on a framework made to simulate routing algorithms in computer networks. Thus, we use the Simulator for Network Algorithms (SinalGo) in this work. SinalGo is a computer network simulation framework developed to test and validate routing algorithms, which has been built with a focus on algorithm verification, abstracting the different layers in a computer network environment. It offers a simple interface to control packet transmission between network nodes, which allows an accurate approximation from the actual operational network hardware.

Fig. 3 shows an example of SinalGo execution, which simulates a network with 144 nodes. Fig. 3(a) means a grid network topology as in [6], but twice in size. In order to build a more realistic experiment, we propose the construction of a totally randomized topology. Fig. 3(b) shows the proposed network used as an experiment. This topology is inherently more difficult for routing algorithms, in general, to construct routes.



(a) Grid network. In with the origin and destination nodes are 1 and 144, respectively marked in blue and red. (b) Fully randomized network. In with the origin and destination nodes are 43 and 119, respectively marked in blue and red.

Fig. 3: Two distinct network typologies simulation with 144 nodes.

Both meta-heuristics addressed in this work have been implemented inside SinalGo using JAVA language. In this framework, each network node emulates a generic router that can run the previously presented algorithms and thus provide routing service. A comparison among the execution times has been carried out in this work. Moreover, three different network configurations are tested to bring the simulation to a real-world problem. That comparison allows us to derive plausible conclusions about the efficiency of each algorithm in a set of environment configurations.

5 Experiments and Results

Each algorithm was executed 30 times for each one of the two environment configurations with 144 nodes each (see Fig. (3) in Section 4). Each configuration represents distinct network structures, the first is a 12x12 grid representation, and the second one represents a totally random network. After each execution, the network is deconstructed and rebuilt, erasing all nodes and connections. This procedure avoids any interference due to the forwarding of routing information from one simulation to another. Before each algorithm execution, a time frame of two hundred events is executed in the network for convergence agents (SA) and network recognition (all methods).

The first step of the three presented algorithms is the discovery phase, which follows a decentralized routing model, with no central element that knows all the network's structure, just those who know their immediate neighbors. In this phase, each node recognizes its neighbors and informs each neighbor of the others. With this data, each node has the information of whatever nodes are reachable through each of its neighbors. Specifically for GA approaches, the crossover operator receives two possible routes, searches for a common node between them, and divides them at this point by generating two new individuals from the combination of the four resulting parts of the division.

The mutation operator receives an individual and searches throughout its length for possible shorter paths between distant nodes, reducing the route's cost and avoiding loops on the network. Initial delay values for each edge in the network is defined as 5 seconds (it's an input necessary to SinalGo). The same delay values are used in all the simulations performed in this work. This range of values is based on the proportion between each edge's transmission and delay times in a network found in the literature [29]. The algorithms have been implemented in JAVA programming language in the SinalGo environment. Table 1 shows the empirical initialization parameters used in all optimization algorithms.

Each algorithm is applied to two distinct network scenarios, where message delivery and delay times, and total time results are accounted for. Statistical inference techniques are applied to analyze the obtained results. Techniques used are boxplot graphics and t-test student [30] (since the result's samples satisfy the normality assumptions). Boxplot is a graphical method to expose numeric data groups through its quartiles. A quartile is a point among the three points

Table 1: Optimization parameter for the algorithms

Genetic algorithm		Simulates Annealing	
Population size	30	Initial temperature T	1000.0
Crossover rate	0.9	Temperature reduction factor α	0.9
Mutation rate	0.02	Repetition schedule M_k	4
Number of generations	30	Maximum number of iterations max_{it}	100

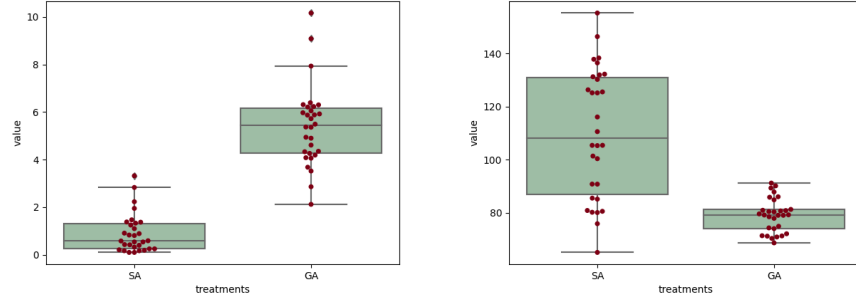
that divides a group of data into four sets with the same amount of elements, one-fourth of the total number of data elements.

The first quartile (Q1) is defined as the median number between the smallest number and the group median (disregarding outliers). The second quartile (Q2) is defined as the median, finally, the third quartile is defined as the median between the group median and the highest values (disregarding outliers) [30]. To perform the comparisons, inference tests, searching to identify the difference between the meta-heuristics used here, are applied. The response variable is the total routing time of the minimal path or, in other words, it is the sum of the time spent to fill out the routing tables of the nodes on the minimum path found.

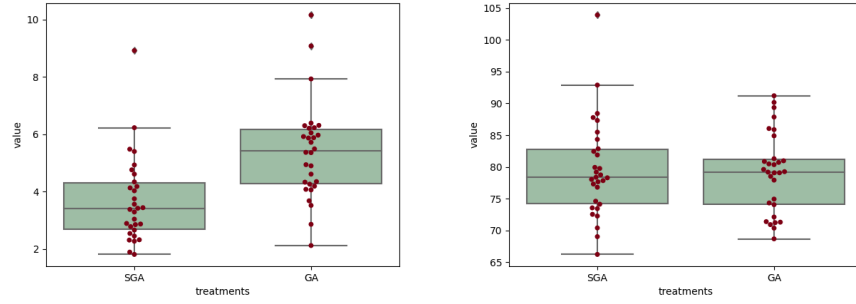
5.1 Grid network topology – Simulation results

Here we are evaluating the algorithms performance in solving the grid network topology, the 12x12 grid with 144 nodes – shown in the Fig. 3(a)). In a visual analysis of the samples of the algorithms, Fig. 4 shows the boxplot for this network. It is possible to notice that SA shows a small time to execute and find a route, as can be seen in Fig. 4(a) when compared to GA. This behavior happens because the SA builds and improves only one solution. At the same time, the GA has to manage and improve a population of solutions, allowing it to enhance those solutions further. When we analyze the total time (Fig. 4(b)) to construct a route and send a packet between origin and destination, it's noticeable that the GA finds and sends packets in a shorter average time.

Moreover, we see that the boxes in Fig. 4(b) do not overlap. This statistically indicates that the null hypothesis of equality of means is refuted. To guarantee this result, a t-student test was performed. The result indicated that with a p-value of 8E-08 the means of the algorithm are statistically different. Since SA builds a solution quickly, but has the disadvantage that it is slower to build routes when compared to GA, we decided to make a coupling between GA and SA (the SGA version). When evaluating time, it is observed that the SGA systematically stops before the GA once there is a stagnation of the final population (Fig. 4(c)). In addition, the average times obtained by the SGA were lower than the GA. This fact indicates that with adjustments to the algorithm parameters, the SGA tends to get better times (Fig. 4(c,d)).



(a) Time for each algorithm execute and find a route. (b) Time for each algorithm execute, find a route and send a packet between origin and destination.



(c) Time for each algorithm execute and find a route. (d) Time for each algorithm execute, find a route and send a packet between origin and destination.

Fig. 4: Grid network topology: boxplot analysis of results

5.2 Fully randomized network topology – Simulation results

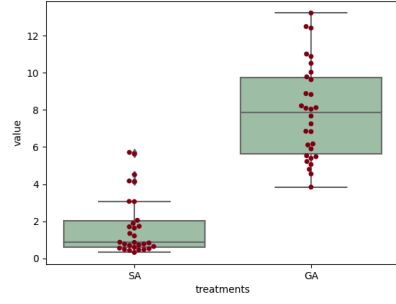
This test case aims to demonstrate the algorithms' behavior on a reduced-scale worldwide network composed of small intra-AS computer networks. This topology can facilitate communication, the sharing of information, and much more between devices from around the world through connected routers. In this experiment, we are simulating a totally randomization network, composed of 144 network nodes (which exemplify routers). This network is shown in Fig. 3(b). We can observe that, in this more complex network, the results showed a similar behavior to the grid network. Fig. 5(a) shows that SA needs less time in the execution and construction phase of an optimized route.

Note that here, the time is almost double compared to the grid network scenario. This is because the nodes are more dispersed and, in some cases, clustered. However, in the packet sending phase, it can be noted that GA has a slightly shorter time, as shown in Fig. 5(b). The boxes do not overlap, which indicates that the hypothesis of equality of means can also be refuted. A student t-test confirmed this statement, since the p-value found was $7E-04$.

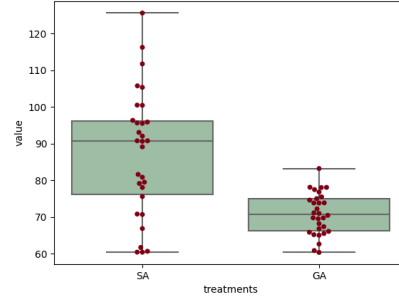
A non-overlapping boxes can be noted among the boxes in Fig. 5(c,d). Here the observed times indicate that the SGA finds an optimized route slightly faster than the GA (see the line that marks the median value in Fig. 5(c)). This is due to the stagnation of the final population. However, no statistically significant difference was observed between the GA and SGA approaches. As future work we indicate a study of parameter setting for the coupled SGA technique. Possibly by adjusting the algorithm, the estimated time for building a route and sending packets from source to destination tends to be shorter.

6 Conclusion

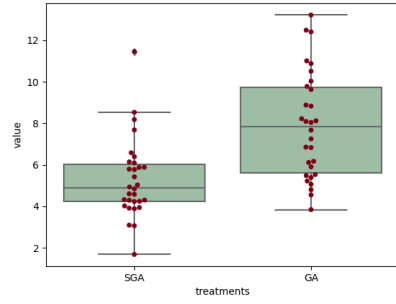
The efficiency, simplicity, and robustness that link-state algorithms have performed over the last decades corroborate the use of these routing algorithms for IP packet traffic in backbone networks. This work has addressed the use of meta-heuristics to solve the problem of the minimal path in packet networks: the Genetic Algorithm with Dijkstra mutation, the Simulated Annealing technique, and coupled algorithm version based on the first two mentioned, using a new simple simulation model approach. These meta-heuristics were compared using two distinct case scenarios with 144 nodes in the network. The experiments indicated that SGA showed competitive results in the presented cases when compared to standard ones. This fact can be justified by the search acceleration in the optimization space done by the SA over the GA proposed. Another advantage is the use of the proposed mutation in GA, which employs the Dijkstra algorithm (link-state one) during the optimization process. The idea of conceiving and developing algorithms based on the evolutionary mixed link-state algorithms proved very plausible based on the data collected. This proposal of new mixed-algorithms can be a solution applicable to the actual demand in the context of processing packets in simulations of small intra-AS computer networks. In this work, we compared three metaheuristics in the framework SinalGo which proved



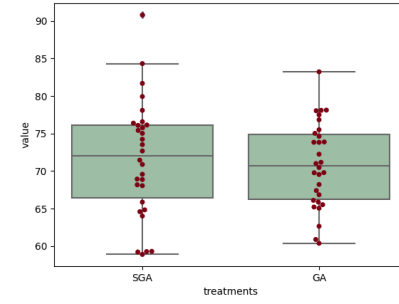
(a) Time for each algorithm execute and find a route.



(b) Time for each algorithm execute, find a route and send a packet between origin and destination.



(c) Time for each algorithm execute and find a route.




(d) Time for each algorithm execute, find a route and send a packet between origin and destination.

Fig. 5: Fully randomized network topology

to be a potent simulation tool. For future outlook, we highlight the experimentation with other goals, such as implementing different technics in SinalGo for comparison, maximizing the quality of service, minimizing weights in link-state routing algorithms, and experimenting with a multi-objective approach.

Acknowledgment

 This project has received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 754382. This research has also been partially supported by Comunidad de Madrid, PROMINT-CM project (grant ref: P2018/EMT-4366) and by the project PID2020-115454GB-C21 of the Spanish Ministry of Science and Innovation (MICINN). The authors thank UAH, UFRJ and CEFET-MG for the infrastructure, and Brazilian research agencies for partially support: CAPES (Finance Code 001), FAPERJ, FAPEMIG, and National Council for Scientific and Technological Development – CNPq. “The content of this publication does not reflect the official opinion of the European Union. Responsibility for the information and views expressed herein lies entirely with the author(s).”

References

1. F. Fazli and M. Mansubassiri. V-RPL: An effective routing algorithm for low power and lossy networks using multi-criteria decision-making techniques. *Ad Hoc Networks*, 132:102868, 2022.
2. Yao Y. and Cao Q. and Vasilakos A.V. Edal: An energy-efficient, delay-aware, and lifetime-balancing data collection protocol for heterogeneous wireless sensor networks. *IEEE/ACM Transactions on Networking*, 23:810–823, 2015.
3. Anwar N. and Deng H. Ant colony optimization based multicast routing algorithm for mobile ad hoc networks. *IEEE Advances in Wireless and Optical Communications (RTUWO)*, 1:62–67, 2015.
4. R. Yadav, S. Indu, and D. Gupta. Review of evolutionary algorithms for energy efficient and secure wireless sensor networks. In K. Khanna, V. Estrela, and J. Rodrigues, editors, *Cyber Security and Digital Forensics*, pages 597–608, Singapore, 2022. Springer Singapore.
5. A. Rovira-Sugranes, A. Razi, F. Afghah, and J. Chakareski. A review of ai-enabled routing protocols for uav networks: Trends, challenges, and future outlook. *Ad Hoc Networks*, 130:102790, 2022.
6. A. Lopez and D.R. Heisterkamp. Simulated Annealing Based Hierarchical Q-Routing: A Dynamic Routing Protocol. In *2011 Eighth International Conference on Information Technology: New Generations*, pages 791–796, 2011.
7. A. Rovira-Sugranes, F. Afghah, J. Qu, and A. Razi. Fully-Echoed Q-Routing With Simulated Annealing Inference for Flying Adhoc Networks. *IEEE Transactions on Network Science and Engineering*, 8(3):2223–2234, 2021.
8. H. Wang, K. Li, and W. Pedrycz. A routing algorithm based on simulated annealing algorithm for maximising wireless sensor networks lifetime with a sink node. *International Journal of Bio-Inspired Computation*, 15(4):264–275, 2020.

9. L. Zhao, A. Saldin, J. Hu, L. Fu, J. Shi, and Y. Guan. A novel simulated annealing based routing algorithm in f-sdns. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1202–1207, 2020.
10. J.S. Raj and K. Rahimunnisa. Hybridized genetig-simulated annealing algorithm for performance optimization in wireless ad-hoc network. *Journal of Soft Computing Paradigm*, 1(3):1–13, 2019.
11. R. Sundar and A. Kathirvel. Aggressively delivered mechanism over variable length density using a simulated annealing algorithm in mobile ad hoc network. *Transactions on Emerging Telecommunications Technologies*, 31(12):e3863, 2020.
12. Prasad A.Y. and B. Rayanki. A generic algorithmic protocol approaches to improve network life time and energy efficient using combined genetic algorithm with simulated annealing in manet. *International Journal of Intelligent Unmanned Systems*, 8(3):23–42, 2020.
13. A. Hamed. A genetic algorithm for finding the k shortest paths in a network. *Egyptian Informatics Journal*, 11:75–79, 2010.
14. L. Zhang, L. Cai, M. Li, and F. Wang. A method for least-cost qos multicast routing based on genetic simulated annealing algorithm. *Computer Communications*, 32:105–110, 2009.
15. A. Younes. Multicast routing with bandwidth and delay constraints based on genetic algorithms. *Egyptian Informatics Journal*, 312:107–114, 2011.
16. A. Bhardwaj and H. El-Ocla. Multipath routing protocol using genetic algorithm in mobile ad hoc networks. *IEEE Access*, 8:177534–177548, 2020.
17. C. Wang, X. Liu, H. Hu, Y. Han, and M. Yao. Energy-efficient and load-balanced clustering routing protocol for wireless sensor networks using a chaotic genetic algorithm. *IEEE Access*, 8:158082–158096, 2020.
18. N. Muruganantham and H. El-Ocla. Routing using genetic algorithm in a wireless sensor network. *Wireless Personal Communications*, 111:2703–2732, 2020.
19. M. Singh, S. Amin, and Choudhary. Genetic algorithm based sink mobility for energy efficient data routing in wireless sensor networks. *AEU - International Journal of Electronics and Communications*, 131:1–10, 2020.
20. E. Heidari, A. Movaghar, H. Motameni, and B. Barzegar. A novel approach for clustering and routing in wsn using genetic algorithm and equilibrium optimizer. *International Journal of Communication Systems*, page e5148, 2022.
21. L. Chu-hang, W. and Xiao-li, H. You-jia, H. Huang-shui, and W. Sha-sha. An improved genetic algorithm based annulus-sector clustering routing protocol for wireless sensor networks. *Wireless Pers Commun*, 123:3623–3644, 2022.
22. A. Bhardwaj and H. El-Ocla. Multipath routing protocol using genetic algorithm in mobile ad hoc networks. *IEEE Access*, 8:177534–177548, 2020.
23. E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
24. H. Chen and B. Sun. Multicast routing optimization algorithm with bandwidth and delay constraints based on ga. *Journal of Communication and Computer*, 2:63–67, 2005.
25. K. Sorensen and F.W. Glover. Metaheuristics. *Encyclopedia of Operations Research and Management Science*, 1:960–970, 2013.
26. S. Stockt and A. Engelbrecht. Analysis of selection hyper-heuristics for population-based meta-heuristics in real-valued dynamic optimization. *Swarm and Evolutionary Computation*, 43:127–146, 2018.
27. A. Nikolaev and S. Jacobson. Simulated annealing. In *Handbook of metaheuristics*, pages 1–39. Springer, 2010.

28. D. E. Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, 3:287–297, 1999.
29. R. Kumar and M. Kumar. Exploring Genetic Algorithm for Shortest Path Optimization in Data Networks. *Global Journal of Computer Science and Technology*, 10:1–5, 2010.
30. D. Montgomery and G. Runger. *Applied statistics and probability for engineers*. LTC, 2009.