Relay selection in Bluetooth Mesh networks by embedding genetic algorithms in a Digital Communication Twin

Jorg Wieme
imec - IDLab
Ghent University
Ghent, Belgium
jorg.wieme@ugent.be

Mathias Baert
imec - IDLab
Ghent University
Ghent, Belgium
mathias.baert@ugent.be

Jeroen Hoebeke
imec - IDLab
Ghent University
Ghent, Belgium
jeroen.hoebeke@ugent.be

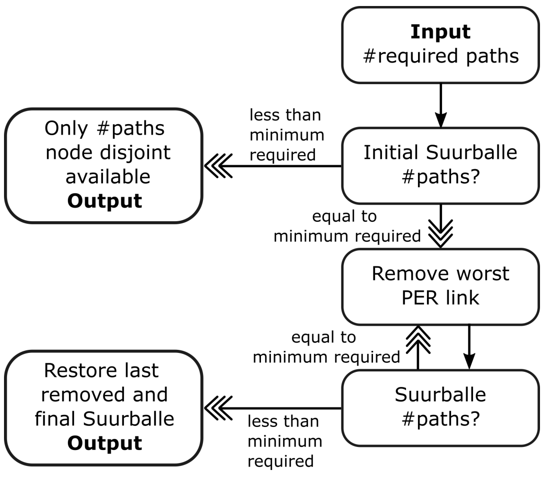
Abstract—**Bluetooth Mesh (BM) technology is a suitable candidate to realize mesh networks leveraging on a mains-powered backbone. However, the flooding-based technology requires additional management in order to operate efficiently. Consequently, it offers a diverse set of configuration options to control network behavior, including the selection of relays in the backbone that rebroadcast packets further into the network. In the past, Digital Twin technology has been applied to design a Digital Communication Twin (DCT) of a BM network. The DCT (or Digital Twin Network, DTN) can be used to find an optimal network configuration for given application requirements. This includes a relay selection approach that finds a set of relays to assure sufficient path redundancy in the network. However, existing results prove that this approach can be improved in terms of computation time, flexibility and validity of the proposed set of relays. In this paper, we present a reinterpreted version of a genetic algorithm to tackle this issue. The results show it achieves better results on all three improvement targets, compared to the original approach. Furthermore, it showcases the flexibility and adaptability of the DCT whilst adequately improving the relay selection approach.**

Index Terms—**Digital Communication Twin, DCT, Bluetooth Mesh, Digital Twin Network, DTN, multipath routing, genetic algorithm, relay selection**

I. Introduction

Bluetooth Low Energy (BLE) can establish single-hop connection-based or connectionless communication between a single master/broadcaster and multiple workers/receivers. The recent extension to Bluetooth technology, Bluetooth Mesh (BM), sets up a multi-hop mesh network, consisting of BLE-enabled devices that employ a connectionless approach to communicate with each other [1]. Devices in a BM network should always be active, to be able to receive and transmit messages at any time [2]. Due to the necessity of a 100 % duty cycle, the base applications for these networks are mains-powered solutions (e.g. smart lighting). The BM specification defines additional concepts to add low-power devices to the network, by leveraging on temporary storage capabilities of the mains-powered backbone. Each node in the network also maintains a repeat value related to the amount of times a source node rebroadcasts a packet after its initial broadcast. Next to this, to ensure that each device in the network can reach other devices related to its application flows, a selected set of nodes in the network should rebroadcast packets (i.e. act as relays). Nodes that act as relay maintain a second repeat value related to the rebroadcasting of relayed packets. The configuration of these per-node parameters should ensure that the network can adhere to the requirements of all application flows and limit the negative impact of them on each other. The work in [3] allows each node to employ a dedicated configuration of these parameters per application flow and can enforce a different priority between flows. This way, the search for an optimal configuration can be performed separately per application flow, but should still take into account the impact of network congestion introduced by other flows. The work in [4] searches for an optimal configuration of a wireless lighting application flow, by means of a Digital Communication Twin (DCT) (can also be referred to as Digital Twin Network or similar) of a BM network. The proposed DCT is a combination of multiple popular approaches for wireless network management, where theoretical models are crossed with simulations into one versatile multifaceted entity that is continuously linked to the physical network. The physical network is enhanced with monitoring mechanisms to provide the DCT with continuous information about the network's state. Currently, the multi-faceted DCT offers a selective simulation component and graph model, both enhanced via a Packet Error Rate (PER) model based on continuous monitoring in the physical network. These facets are used to gain insights in the network or to find more optimal configurations. For this, a flexible approach has been taken in the form of recipes and pipelines thereof. With the recipe-controlled pipeline it is possible to configure the network partially or completely matching the application input requirements, through the usage of one recipe or a chain of recipes, which is visualized in Fig. 2. A recipe typically focuses on optimizing one aspect of the network (i.e. path redundancy, Packet Delivery Ratio (PDR), End-to-End (E2E) latency, etc.) or assessing network performance based on the current configuration or other reconfiguration suggestions. The connectivity recipe focuses on path redundancy, which has the goal to guarantee a reliability baseline based on a redundancy target, whilst maintaining the lowest number of relays. This directly impacts the number of rebroadcasts, thus reducing the network congestion over time. Removing too many relaying nodes would drastically decrease the reliability of the network. Consequently, the recipe searches for an optimal balance between the redundancy and its side effects, as depicted in Fig. 1. The recipe starts by performing Suurballe [5] for the redundancy target as number of node disjoint paths to find. The algorithm uses hop count as weight, thus every arc has cost 1. Once this is computed, the highest PER link is removed from the graph, and Suurballe is computed again. This process is repeated until Suurballe fails to compute the requested number of node disjoint paths. Finally, the last removed link is restored, and the recipe provides the outcome of the Suurballe algorithm for the current topology as output. This can be used as an input parameter for subsequent recipes in the pipeline.

The current connectivity assessment provides valid paths but distinguishable issues occurred during our analytical evaluation. The DCT must be responsive in order to provide insights into a rapidly changing network environment. The responsiveness of the recipe is within reasonable range but it would be beneficial to have an even lower computation time. Next, it was proven in [4] that the distributions of the simulation results align with the experiments in the physical network. However, when performing the simulation and experiments with the relay configuration provided by the connectivity assessment recipe, the paths taken are always shorter than the suggested paths by the recipe. This issue arises through the restrictive concept of the recipe where it solely focuses on finding node disjoint paths with links having the lowest PER link in the network, by subsequently removing the worst PER link without taking into account the number of hops (except using it as to be minimized weight in the Suurballe algorithm). This leads to longer paths, while in reality, the path distributions of the results prove that the number hops have a much stronger impact on the path that will be taken, even though the link shortcuts on those longer paths have a slightly higher PER. Those E2E path distributions prove that it is possible to define a smaller set of relaying nodes that achieves the same reliability baseline while providing a more truthful representation of the E2E path distribution in the physical network. The link shortcuts on the longer paths and crosslinks between the different paths are not necessarily a bad thing since they are part of the nature of a flooding-based approach. But the current outcome of the recipe introduces to much unnecessary overhead by relays that do not contribute to the successful E2E communication between source and destination. Our goal is to limit the number of relays as much as possible by means of a different connectivity assessment approach. The paths selected by the recipe should be, as much as possible, the dominating paths in the distributions while still benefiting from the inherently present link shortcuts and crosslinks. Lastly, as mentioned, internally the recipe uses Suurballe to find node disjoint paths, which works appropriate in a dense network, but it is not applicable in a sparse network, implicating that the recipe is not versatile enough. This paper continues to build upon the proposed DCT solution in [4], by assessing some improvements for the connectivity assessment recipe and designing a replacement for it by means of a reinterpreted way of a genetic algorithm, illustrating the flexibility and adaptability of the DCT. The original recipe and its replacement are then evaluated by focusing on the computation time, flexibility and a finding a lower set of relays to adhere to a given path redundancy target. The genetic algorithm (and by extension, the entire DCT) are described for a BM network, but the concept is applicable to other technologies as well.



Input
#required paths
Initial Suurballe #paths?
less than minimum required
Only #paths node disjoint available
Output
equal to minimum required
Remove worst PER link
equal to minimum required
Suurballe #paths?
less than minimum required
Restore last removed and final Suurballe
Output

Fig. 1. Workflow diagram for the connectivity assessment recipe.

The remainder of the paper is organized as follows. Section II describes some extensions for the original connectivity assessment recipe. Section III provides a replacement for this recipe by means of a genetic algorithm. Section IV compares the performance of the novel genetic algorithm with the connectivity assessment recipe. Finally, Section V provides a conclusion and an outlook on future work.

II. Improving the Original Recipe

A. Minimizing Error Rather Than Hop Count

Currently, hop count is used as weight in the Suurballe algorithm. However, in [6] it was experimentally proven that minimum-hop-count routing is not a reliable metric for wireless multi-hop networks as it often chooses significantly weaker paths. Instead, the work in [7] proposed Expected Transmission Count (ETX) as metric. Other works such as [8], [9], [10] compared and designed different metrics for wireless network routing. However, none of these proposals are applicable for us because acknowledgment packets are required. Instead, we can extend the recipe with the option to instruct Suurballe to minimize E2E PER instead of number hops. Suurballe [5] is conceptually built upon Dijkstra's shortest path algorithm [11]. Dijkstra's shortest path algorithm uses summation of arcs, thus the arc values for PER are not immediately suitable as Dijkstra weights. Instead, we

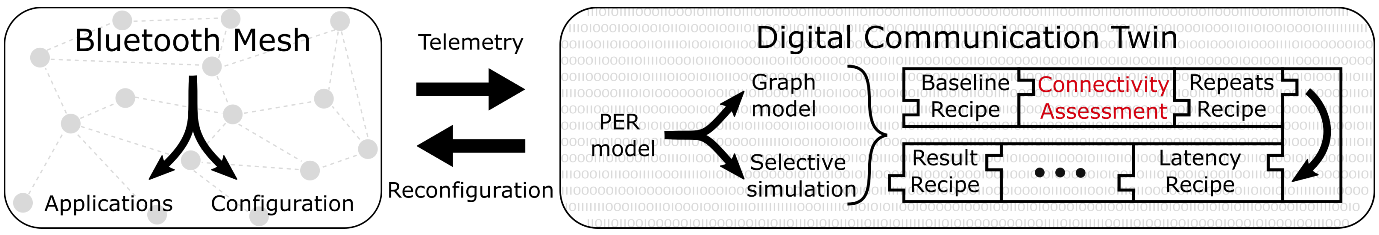


Fig. 2. Illustration of a DCT linked to a BM network. It can provide insights via a single recipe or assess an entire application by means of a recipe pipeline.

propose the following equation that can be used by Dijkstra (& Suurballe) to minimize E2E PER:

$$\begin{aligned} \textit{Prob[No error on path]} &= \prod_{i=1}^{N}(1 - PER_i) \\ \textit{Prob[Error]} &= 1 - \prod_{i=1}^{N}(1 - PER_i) \\ &- \log\left[\prod_{i=1}^{N}(1 - PER_i)\right] \textit{(minimize)} \\ &\sum_{i=1}^{N}\log\left(\frac{1}{1 - PER_i}\right) \\ &\log\left(\frac{1}{1 - PER_i}\right) \textit{(arc weight)} \end{aligned} \tag{1}$$

To compute the E2E packet success ratio (PSR) of a path, the $(1-PER)$ of all arcs on that path are multiplied, as defined in (1) on the first line. The opposite of this value is the E2E PER, which should be minimized. Similarly, the logarithmic value should also be minimized. Thus, if we apply the logarithmic value and then use its corresponding mathematical rules, we achieve a summation that can be employed by Dijkstra. Conceptually, this formula is equivalent to ETX without the reverse delivery ratio. Nevertheless, using (1) as metric in the recipe did not result into an improvement, for both sparseness as well as the number of relays. This occurs because we are still using the same idea of removing the worst PER link in between each usage of Suurballe, thus the result given by each iteration of Suurballe differs, but in the end the recipe does not provide a better result.

B. Preemptive Filtering

A different approach conceptualized was the usage of filters to reduce the search space. We first collect all the possible paths for the current network topology. Once collected, the recipe is applied on the 10% best paths, where the best is quantified using the E2E PSR. Applying this filter ensued into a set of paths that better aligns with the simulation results, due to the fact that the initial filtering of paths limits the number of worst PER links to be removed by the connectivity assessment recipe and allows the Suurballe algorithm to operate directly in a reduced, more probable search space. However, the filtering mechanism is infeasible to run within reasonable time for larger networks.

III. Genetic Algorithm

A. Classical Interpretation

In Section II we examined some extensions to the existing recipe but this did not provide sufficient improvements regarding computation time and finding a smaller set of relays. This section proposes an entirely different approach, by means of a genetic algorithm. The connectivity assessment problem is reducible to a k-shortest path problem from source to destination, with an additional rule for node disjointness. Various research has been conducted to solve the k-shortest path problem in a network using genetic algorithms [12], [13], [14], but they did not apply a node disjoint restriction on the results. Other research works ([15], [16]), that do not rely on a genetic algorithm, did apply the node disjoint restriction on their results but are not within responsive time and often required unavailable information. The genetic algorithm proposed by [17] did solve k-shortest path whilst maintaining a reduced form of node disjointness. The node disjoint restriction is substituted into an independent rule, where the recurrence of nodes is allowed but highly discouraged. For a sparse network it is often impossible to get k node disjoint paths, hence the relaxation of the node disjoint restriction. Initially, they proposed to solve the k-shortest path problem using a genetic algorithm and keeping track of the k-best results throughout all generations. They concluded that most of those paths are a clone of the best path with some minor variations. Thus, their second proposal was to use multiple fitness functions, where each fitness function corresponds with some geographical location. This way, each fitness function searches for the best path that correlates to a different section of a map. Unfortunately, BM networks do not have geographical information for each device. Additionally their solution only supports one metric for path length, either hop count or path cost, whilst we require a combination of those. Hence, a different approach is proposed with our genetic algorithm.

B. Design of a Nucleotide based Genetic Algorithm

Genetic algorithms consist of a population, where a population is a collection of chromosomes, this population evolves across multiple generations [18]. We introduce a new term named the nucleotide. The remainder of this subsection explains how every aspect of our genetic algorithm is set up.

Nucleotide: A nucleotide represents one possible path in the entire network, this representation is equal to the chromosome definition in [17]. All nucleotides have the same length, with the length being equal to the number of nodes in the network. Thus, a fixed array is used and in this array the index represents the source of an unnamed arc, whilst the value at that index indicates the destination of that arc. Chaining this concept, starting at the source node until the destination node, allows us to store a path. The indices that are not on the path contain randomly one of their neighbors.

Chromosomes: For this algorithm the chromosome is a collection of nucleotides, and it contains exactly k nucleotides. Therefore, one chromosome could be a possible solution for the k-shortest path problem. The population remains equal to the common interpretation, where it contains a certain number of chromosomes and each generation the population will evolve.

Fitness Function: Fitness functions are defined for both the nucleotide and the chromosome. As with the nucleotide definition, the fitness function of the nucleotide is inspired by the fitness function of the chromosome in [17]:

$$Fitness_N = \frac{1}{\sum_{i=1}^{N} arc(i)} \bigg/ \#hop \tag{2}$$

with N representing the number of nodes on a path and the arc is calculated as in (1) where node i is the destination of the arc, and $i-1$ the source. However, as previously illustrated, it is not sufficient to focus solely PER, i.e. the hop count has a significant impact on the path taken, due to the nature of flooding-based communication. Thus, the fitness function does not compute the shortest path, but the shortest path on average across the hop count. This way we mix the PER with the hop count, and therefore prioritize short and relatively strong probable paths over long and very probable paths. Due to the division, a higher fitness value implies a shorter path with better E2E PSR. The fitness value of a chromosome is computed as follows:

$$Fitness_C = \frac{(\#uniq - 2) \times \sum_{k}^{i} Fitness_N(i)}{(\#total - k * 2)} \tag{3}$$

$Fitness_N$ from (2) is calculated for each nucleotide in the chromosome. Path independency (i.e. disjointness) is promoted by punishing recurring nodes across all nucleotides. Recurring nodes are defined by $\#uniq/\#total$ where $\#uniq$ represent the number of unique nodes within one chromosome and $\#total$ is the number of nodes in one chromosome. However, to uniformly define the denominator, we need to remove the source and destination from the equation.

Crossover: Half of the crossovers are applied on the nucleotide level whilst the other half is applied on the chromosome level. Lethal nucleotides are suppressed in similar fashion as crossover defined in [17]. Chromosome crossover is significantly easier when nucleotides are randomly exchanged between two selected parents. No lethal chromosomes can be composed using this technique.

C. Usage of a Nucleotide based Genetic Algorithm

Generation of the nucleotides and chromosomes is completely random. Consequently, a nucleotide is created as follows. Starting at the source, pick a neighbor. This is now temporarily the source. Repeat this process until the neighbor is the destination. If no neighbors are left, discard the nucleotide and restart the creation process from scratch. A chromosome is formed by performing the previous operation k times. The selection process uses a spinning biased roulette wheel, similar to [17]. However, our replacement strategy is not elitism [17] but rather fitness-based selection where the offspring is combined with its parents into the next generation. Lastly, the mutation process is naively defined, i.e. instead of making small changes, a completely new nucleotide or chromosome is generated. This decreases the convergence rate but it prevents the local optima problem, which is the intended goal of mutation in genetic algorithms [18].

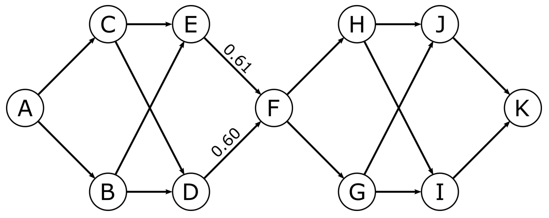


Fig. 3. Representation of a sparse network. Every arc, except $E \rightarrow F$ and $D \rightarrow F$, has an error probability of 1%

IV. Evaluation

The evaluation compares the original connectivity assessment recipe with our proposed solutions, regarding flexibility, computation time and finding a minimal set of relays to achieve redundancy. The evaluation is performed via simulations on a DCT of both an artificial network (i.e. flexibility) and a physical network (i.e. computation time and relay selection), where monitoring mechanisms are present [4]. The physical network represents a BM network in a dynamic office environment, where mains-powered lighting is used as a backbone.

A. Flexibility

For the assessment of the sparseness problem, we created a small artificial network represented in Fig. 3. As can be seen, the network is divided into two subnetworks that are connected through one node. An application flow that starts in the left subnetwork and ends in the right subnetwork must always visit that connecting node. Note that there are other methods, not related to connectivity assessment, to solve this problem, however in this situation it merely serves as an example for sparseness. If the original connectivity assessment recipe is executed with a redundancy target of 2, it provides us with only one path: $A \rightarrow C \rightarrow D \rightarrow F \rightarrow H \rightarrow J \rightarrow K$. In spite of this path being the shortest possible path, one of its arcs has a high error rate, implying that a secondary path could have been appropriate to improve the network reliability. However, the node disjoint restriction simply does not allow this method to compute a secondary path for this network. Executing our genetic algorithm with the same redundancy target, returns

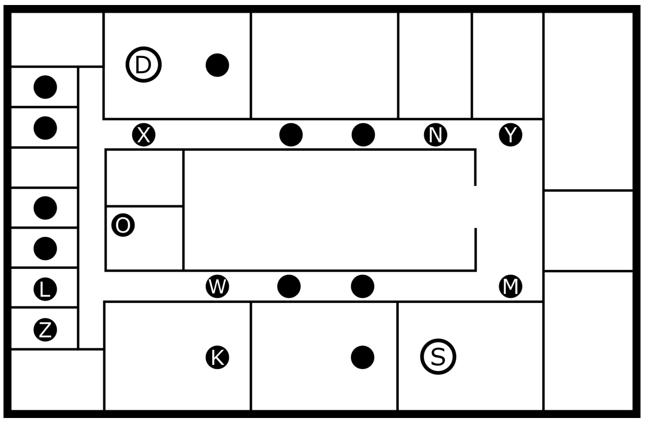


Fig. 4. Testbed representing a BM network.

the requested number of paths, thus including a second path: $A \rightarrow B \rightarrow E \rightarrow F \rightarrow G \rightarrow I \rightarrow K$.

B. Computation Time

The DCT mirroring the physical network is extended with our genetic algorithm as recipe, enabling accurate comparison of the computation time between the original recipe and the proposed replacement. Fig. 4 represents the topology of the physical network in a real-life testbed, i.e. a network containing 21 nodes[1]. It represents an office environment where mains-powered lighting serves as the backbone. Throughout the evaluation we will work with a fixed source and destination, both marked with their respective abbreviation S and D respectively. The standard raw configuration is equivalent for all the nodes, where each node is a relay and uses 0 repeats. The number of possible paths between the source and destination in this network is of an exponential factor. With a maximum path length of 7 hops, there are approximately 2.5×10^5 paths between the source and destination in our network. If this value is extrapolated to the maximal possible hop counts, there are approximately 6.4×10^{16} paths, making the search space non-iterable within reasonable computation time. We calculate the computation time for three versions of the connectivity assessment recipe: the original version from [4], an enhanced version using (1) as arc weight and the genetic algorithm approach. This process was repeated on several snapshots of the DCT, across multiple hours, i.e. to experience, capture and test the volatility of an office environment. The original recipe performed with hop count or formula (1) takes $22.14s$ and $24.42s$ on average respectively. In contrast, the genetic algorithm takes only $1.60s$. This clearly illustrates that the genetic algorithm performs on average ten times faster than the current solution. We also see that the different metric used within the original version does not have any impact on the computation time, which is expected given that both versions use the same workflow as in Fig 1. Each iteration of the original workflow executes the Suurballe algorithm and removes the worst PER link, until the redundancy target is no longer satisfied. This leads to a much longer computation time, compared to our genetic algorithm approach using 1000 generations.

[1]Office Lab testbed at Ghent University

C. Relay selection

The objective of this part of the evaluation is to assess the accuracy of the relation between the suggested paths (i.e. relay selection) by the evaluated recipe and the path distribution of a subsequent run of 200 simulations of the same communication flow, performed on the network depicted in Fig. 4. We assess two versions of the connectivity assessment, i.e. the original approach from [4] and the genetic algorithm. Each outcome represents a set of relays and a simulation is performed for both relay selection sets, providing us with two path distributions, i.e. original & genetic algorithm based. The accuracy of each version is calculated by counting the occurrence of the suggested paths from the recipe in their associated path distribution.

The accuracy assessment is illustrated in Table I, where the path distribution is sorted on the number of occurrences for each path in the simulation, after applying either the original connectivity assessment recipe or the genetic algorithm approach. The original connectivity assessment recipe indicates that there is not a single suggested path in the top 4 most occurring paths. In contracts, the assessment of the genetic algorithm approach indicates that all 3 suggested paths are in the top 4. It is clear that the genetic algorithm provides a more accurate representation of the relation between suggested paths and a simulated path distribution. The process used

TABLE I
SORTED PATH DISTRIBUTION FOR EACH RECIPE VERSION OUTCOME

Version	Path	#	Redundancy
Original	$S \rightarrow K \rightarrow D$	59	NO
	$S \rightarrow L \rightarrow D$	37	NO
	$S \rightarrow M \rightarrow N \rightarrow D$	12	NO
	$S \rightarrow L \rightarrow O \rightarrow D$	11	NO
	...	...	...
	$S \rightarrow K \rightarrow O \rightarrow D$	4	YES
Genetic	$S \rightarrow Z \rightarrow D$	69	YES
	$S \rightarrow Y \rightarrow D$	34	NO
	$S \rightarrow Y \rightarrow X \rightarrow D$	32	YES
	$S \rightarrow W \rightarrow D$	19	YES
	$S \rightarrow Z \rightarrow X \rightarrow D$	15	NO
	...	...	...

in Table I was repeated for several iterations of the two connectivity assessment versions and the associated simulation to find associated path distributions. The average accuracy of this endeavor is 0.76% (with 3.5 suggested relays on average) for the original recipe version and 58.35% for the genetic algorithm (with 1.5 suggested relays on average). As expected from the results in indicated in Table I, the average accuracy of the original solution is very low. The genetic algorithm provides a drastic increase in average accuracy, providing a more truthful assessment on realistic network behavior. The remaining 40% are paths that use link shortcuts on the suggested paths and crosslinks between them. Although our goal is to design an algorithm that provides a relay selection that is as truthful as possible, the usage of these link shortcuts and crosslinks is only natural. This behavior is also present in the original solution, but there the suggested paths are too distinct from realistic network behavior and thus lead to too much unnecessary overhead from useless relays. The genetic algorithm approach has proven to drastically reduce the relay selection set (indicated by the # relays column), which is exactly one of the goals that we want to achieve in this paper.

V. Conclusion and Future Work

In this paper a restructured genetic algorithm was proposed to replace an existing, flawed, solution for a multi-path routing problem. It has been added as an additional recipe to a DCT's toolbox, proving the latter's adaptability towards extensions. Its accuracy, speed and flexibility were experimentally evaluated and compared to the original solution. The results prove that the genetic algorithm can circumvent restrictions of a sparse network (which the original solution was not able to do), has a significant lower computation time and provides higher relay accuracy, leading to much less unnecessary relays and their associated overhead. The fitness function can easily be extended or adapted to similar requirements, making it versatile for other routing restriction problems. Further improvements on each of the evaluated aspects are possible. Speed and convergence rate would increase if additional alterations were applied. For example, a proper mutation operation would increase the convergence rate. Having a better convergence rate would enable a lower number of generations, increasing the speed but also accuracy, by avoiding continuous iterations within a local optima. Aside of conceptual improvements, implementation wise decisions could further improve the speed as well. The connectivity assessment recipe is only one aspect of the recipe pipeline proposed in [4]. Consequently, a comparison should be made between the original pipeline and a enhanced version that replaces the connectivity assessment recipe with the genetic algorithm. Furthermore, it should be examined whether an extension of the genetic algorithm itself with additional parameters corresponding to other recipes in the pipeline, yields better reconfiguration suggestions compared to the original or enhanced pipeline. For example, the goal of the repeats recipe is to further enhance the reliability towards a PDR target by means of additional repeats on the paths provided by the previous recipe. The genetic algorithm could include the number of repeats for both the source node and intermediate relays in the fitness function and find paths according to required redundancy and PDR simultaneously. However, inclusion of too many parameters in a single recipe potentially undermines the original idea of a DCT that combines diverse approaches into a single solution instead of a single model that is continuously extended [4].

References

[1] J. Yin, Z. Yang, H. Cao, T. Liu, Z. Zhou, and C. Wu, "A survey on bluetooth 5.0 and mesh: New milestones of iot," *ACM Trans. Sen. Netw.*, vol. 15, no. 3, may 2019. [Online]. Available: https://doi.org/10.1145/3317687

[2] M. Baert, J. Rossey, A. Shahid, and J. Hoebeke, "The bluetooth mesh standard: An overview and experimental evaluation," *Sensors*, vol. 18, no. 8, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/8/2409

[3] S. S. Basu, M. Baert, and J. Hoebeke, "Qos enabled heterogeneous ble mesh networks," *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, 2021. [Online]. Available: https://www.mdpi.com/2224-2708/10/2/24

[4] M. Baert, E. De Poorter, and J. Hoebeke, *A Digital Communication Twin for Performance Prediction and Management of Bluetooth Mesh Networks*. New York, NY, USA: Association for Computing Machinery, 2021, p. 1–10. [Online]. Available: https://doi.org/10.1145/3479242.3487327

[5] J. W. Suurballe, "Disjoint paths in a network," *Networks*, vol. 4, no. 2, pp. 125–145, 1974. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/net.3230040204

[6] D. S. J. De Couto, D. Aguayo, B. A. Chambers, and R. Morris, "Performance of multihop wireless networks: Shortest path is not enough," *SIGCOMM Comput. Commun. Rev.*, vol. 33, no. 1, p. 83–88, jan 2003. [Online]. Available: https://doi.org/10.1145/774763.774776

[7] D. S. J. De Couto, D. Aguayo, J. Bicket, and R. Morris, "A high-throughput path metric for multi-hop wireless routing," in *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '03. New York, NY, USA: Association for Computing Machinery, 2003, p. 134–146. [Online]. Available: https://doi.org/10.1145/938985.939000

[8] R. Draves, J. Padhye, and B. Zill, "Comparison of routing metrics for static multi-hop wireless networks," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 133–144. [Online]. Available: https://doi.org/10.1145/1015467.1015483

[9] L. Zhao and A. Y. Al-Dubai, *Routing Metrics for Wireless Mesh Networks: A Survey*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 311–316. [Online]. Available: https://doi.org/10.1007/978-3-642-25769-8_45

[10] R. Draves, J. Padhye, and B. Zill, "Routing in multi-radio, multi-hop wireless mesh networks," in *Proceedings of the 10th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 114–128. [Online]. Available: https://doi.org/10.1145/1023720.1023732

[11] E. W. Dijkstra *et al.*, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[12] A. Y. Hamed, "A genetic algorithm for finding the k shortest paths in a network," *Egyptian Informatics Journal*, vol. 11, no. 2, pp. 75–79, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S111086651000023X

[13] G. Nagib and W. G. Ali, "Network routing protocol using genetic algorithms," 2010. [Online]. Available: https://api.semanticscholar.org/CorpusID:9226476

[14] A. Bhardwaj and H. El-Ocla, "Multipath routing protocol using genetic algorithm in mobile ad hoc networks," *IEEE Access*, vol. 8, pp. 177 534–177 548, 2020.

[15] T. Chondrogiannis, P. Bouros, J. Gamper, U. Leser, and D. B. Blumenthal, "Finding k-dissimilar paths with minimum collective length," in *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 404–407. [Online]. Available: https://doi.org/10.1145/3274895.3274903

[16] T. Akiba, T. Hayashi, N. Nori, Y. Iwata, and Y. Yoshida, "Efficient top-k shortest-path distance queries on large networks by pruned landmark labeling," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, ser. AAAI'15. AAAI Press, 2015, p. 2–8.

[17] J. Inagaki, M. Haseyama, and H. Kitajima, "A genetic algorithm for determining multiple routes and its applications," in *1999 IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 6, 1999, pp. 137–140 vol.6.

[18] S. Katoch, S. S. Chauhan, and V. Kumar, "A review on genetic algorithm: past, present, and future," *Multimedia Tools and Applications*, vol. 80, no. 5, pp. 8091–8126, Feb 2021. [Online]. Available: https://doi.org/10.1007/s11042-020-10139-6