# Optimizing Scheduling in Wireless TSN Utilizing Genetic Algorithms

Jetmir Haxhibeqiri*, Pablo Avila-Campos*, Ingrid Moerman*, Jeroen Hoebeke*
*IDLab - imec, Ghent University, Belgium
{jetmir.haxhibeqiri, pabloesteban.avilacampos, jeroen.hoebeke, ingrid.moerman}@ugent.be

***Abstract*—Time-sensitive networking (TSN) is proposed to support deterministic communication for industrial automation use cases. To harvest the wireless communication flexibility, TSN features have been extended to the wireless domain as well. One of the key TSN features is the ability to assign transmission schedules to different traffic flows in the network with the aim of reducing time slot access delay on each network node. In the wireless domain, this becomes even more challenging due to the shared medium, lower reliability, and slower transmission rates compared to wired systems, reducing the available time resources. In this paper we look at utilizing genetic algorithms to support scheduling of traffic flows from different wireless end devices in a shared schedule. Two optimization functions are defined. The first optimization is based on minimizing the overall shared air time between different end nodes, while the second optimization is based on maximizing time slots that can be used without any interference. Both optimizations aim to reduce the collision probability. With these initial results, we identify the best parameters for genetic algorithms and examine the initial population's impact on overall performance. We show that a fully randomized initial population does not achieve the highest fitness value, even after several generations.**

***Index Terms*—wireless TSN, scheduling, genetic algorithms**

## I. Introduction

In a robotic assembly line, several robots perform tasks like picking, placing, welding, and painting components to assemble products. They must communicate with each other and the central control system in real-time to coordinate movements precisely. Sensors throughout the line give feedback on component status, environmental conditions, and safety hazards. Such interaction between robots and sensors ensures efficient and safe operation. Such precise coordination relies on meticulous management of time-sensitive communication flows among network applications. Time-sensitive networking (TSN) has emerged as a solution to advance current networks to support this deterministic communication and continues to attract significant attention from both academia and industry.

Currently, the TSN Working Group, which started as the Audio Video Bridging (AVB) working group back in 2012, has introduced several standards. These standards include the IEEE 802.1AS for time synchronization, IEEE 802.1CB for reliability, IEEE 802.1Qbv for time-aware shaping (TAS) scheduling etc. Certainly, the current standards predominantly focus on wired TSN implementations. However, eliminating wired connections between robots, sensors, and the control system, would not only facilitate mobility and plug-and-play capabilities but also enhance communication flexibility and efficiency. These advancements are crucial in realizing the *smart factory* vision of Industry 4.0, where a combination of wired and wireless TSN (W-TSN) provides seamless integration, adaptability, and optimization [1].

While the TSN standards offer mechanisms to manage and prioritize time-sensitive traffic flows, they lack a scheduling mechanism capable of delivering the latency and jitter guarantees promised by TSN. Indeed, scheduling approaches are use-case specific and cannot be standardized. As such, to address this deficiency, various algorithms have been proposed in the literature based on different assumptions and optimization objectives to fulfill diverse constraints. However, many of these works overlook the specific wireless TSN, such as lower transmission rates and reliability compared to wired networks, as well as channel variability in time and frequency, which necessitates continuous adaptations, for instance, through Modulation and Coding Schemes (MCS) index dynamic selection.

A widely used strategy for addressing the scheduling problem begins by establishing different traffic flow priorities based on their requirements. Typically, three classes are proposed: high time-sensitive (HTS), which corresponds to time-aware cyclic and latency-bounded streams; low time-sensitive (LTS), corresponding to control and configuration traffic, such as time synchronization, which is acyclic but still latency-bounded; and finally, best effort (BE) traffic, representing acyclic and non-latency-bounded traffic [2]. While HTS traffic is normally supported by selected dedicated time slots in the communication cycle and over-provisioning of air-time for the flow, the other two traffic flows still should be supported in the same network. This work aims to optimize time resource sharing to maintain acceptable time-sensitive advantages for LTS and BE traffic in W-TSN. Optimization of the air-time resources between different wireless end devices and traffic flows relies on the learning and adaptability of genetic algorithms.

The structure of the paper is outlined as follows. Section II discusses related work on scheduling while section III provides background information on genetic algorithms. Section IV describes the optimization problem, while section V details the simulation environment. Section VI presents achieved results, while section VII gives concluding remarks.

## II. Related Work

TSN scheduling can be broadly categorized into exact scheduling methods and approximate scheduling methods.

Exact methods encompass techniques like Integer Linear Programming (ILP), Satisfiability and Optimization Modulo Theories (SMT/OMT) [3], and constraint programming. In contrast, approximate methods comprise heuristics, artificial intelligence (AI), and genetic algorithms [2].

Scheduling traffic flows in wired TSN is a well-established problem that has garnered significant interest from the research community in the past. In [4], a method called Greedy Randomized Adaptive Search Procedure (GRASP) has been applied to solve the gate control lists (GCL) solution problem when combining HTS and LTS traffic types. This solution focuses on worst-case end-to-end delays and demonstrates that an optimal combination of transmission time slots between HTS and LTS, may reduce the worst-case end-to-end delay. In [5], authors present the no-wait packet scheduling solution for GCLs when time-aware shaping is applied. While some works propose only scheduling, others, such as [6], include routing in the optimization problem by introducing an objective function that incorporates the waiting time of flows and flow schedulability. Similarly, authors in [7] address routing and scheduling in a multi-hop wired TSN as an iterated integer linear programming problem.

While ILP is a well-studied technique for TSN scheduling in wired networks, formulating the problem can be complex, and it often requires substantial computational time to obtain a solution. Therefore, to overcome this challenge, techniques based on machine learning (ML) algorithms are gaining traction. In [8], authors present a scheduling method that is based on a deep reinforcement learning (DRL) algorithm improving the efficiency and schedulability of traffic flows for more than 35% compared to heuristics. However, ML-based scheduling methods still require time to train the model, or in the case of DRL, there is always a trade-off between the exploration and exploitation phases. Other works have employed genetic algorithms for the scheduling problem as an approximate method only for wired TSN [9].

Compared to wired TSN, in W-TSN network dynamicity is not composed only by the different traffic flows but also due to node mobility and utilization of dynamic MCS. This poses an additional challenge in the scheduling of wireless links. In [10], authors propose a cross-traffic scheduling, where inbound traffic shaping is performed based on frame length shaping, while the outbound traffic scheduling is done based on queue status. In [11], network dynamism is emphasized in the scheduling problem attributed to changes in the number of traffic flows.

## III. Background to Genetic Algorithms

Genetic algorithms are based on natural selection and genetics that search through all the possibilities of the solution space, evolving from less optimal solution towards optimal or near-optimal solutions. In order to explain how genetic algorithms work, first we will list a number of notions that will be used. A subset of all possible solutions is recognized as *population*. *Initial population* is the subset of solutions from which optimization starts. One solution in the population represents a *chromosome* that is composed of several *genes* taking several possible values. Lastly, a *fitness function* is defined to determine how close to an optimal solution a solution is. The *fitness function* takes a *chromosome* as input and gives the *chromosome* suitability as output [12]. In order for a population to evolve, genetic operations need to be performed in the initial and subsequent populations, such as: parent selection, crossover, mutation, survivor selection, etc.

The first step in genetic algorithm is to calculate the fitness function of each *chromosome* in the *population*. The fitness function should be easy to calculate in order to speed up the search process. A selected set of parents (chromosomes) will mate and recombine in the second step to generate the new chromosomes for the upcoming generation. Usually, the parents are selected from chromosomes with the highest fitness value. Several techniques are used for parent selection, such as roulette wheel selection, $k$-way tournament selection, rank selection, or even random selection technique.

The third step in the genetic algorithm is the crossover process, where from more than one parent one or more offsprings are generated. Similarly, in crossover step a number of techniques can be used, like the one-point or multi-point crossover, uniform crossover, arithmetic recombination, etc.

The fourth step in the genetic algorithm is mutation. Mutation represents a small tweak of *genes* in *chromosomes* happening with a low probability. The last step is survivor selection which determines which chromosomes need to be removed from the next generation and which will be passed from the current generation to the other.

## IV. Optimization problem

### A. Shared vs dedicated time slots in schedules

Scheduling in W-TSN involves controlling each node's channel access, which includes transmission queues and a gated mechanism. In the case of W-TSN, the gated mechanism must be synchronized across devices due to the shared wireless medium. In this work, we assume that W-TSN utilizes standard IEEE 802.11 with this gated mechanism implemented on top.

Wireless transmissions have lower data rates than wired ones. Therefore, a smaller communication cycle improves time slot access delay but requires more shared air-time for different traffic flows. However, due to limited air-time space, not all queues can have dedicated time slots. Queues for LTS and BE traffic flows often use shared time slots, where contention can increase because of the TSN schedule. Even with a gated mechanism, nodes must follow the IEEE 802.11 listen-before-talk (LBT) protocol. If two nodes try to access the channel at the beginning of a shared time, they will start distributed coordination function (DCF) interframe spacing (DIFS) simultaneously. Consequently, this results in simultaneous transmission commencement, leading to packet collision, increasing contention within the time slot, and wasting valuable air-time as seen in Figure 1.

To avoid this issue, time slots for LTS traffic can start at different times, having different shared air-time amounts [13]. By shifting the start times, contention can be optimized within

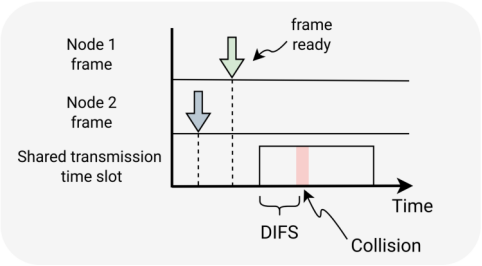



Fig. 1: Shared time slot contention

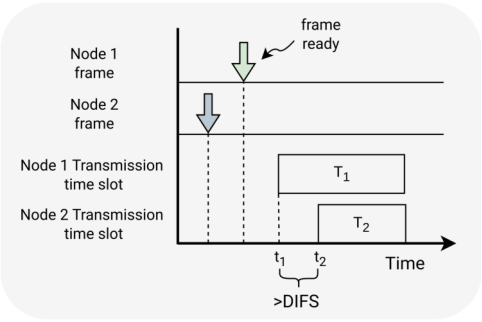



Fig. 2: Share time slot prioritization

the same shared air-time space for all LTS and BE traffic flows. Unlike shared time slots, dedicated time slots are assigned to a single traffic flow, typically used for HTS traffic. A node needs reserved channel access at least at the beginning of the time slot to ensure dedicated access. Once the transmission starts, other nodes will defer due to the LBT mechanism as shown in Figure 2.

### *B. Optimization*

First, we define the variables and constraints that would be considered in the scheduling problem. Let denote with *C* the TSN communication cycle length, containing the set *T* of all *n* time slots $(T_1, T_2, \ldots, T_n)$ for *n* different LTS and/or BE traffic flows initiated by different wireless nodes. Each time slot can have a different length that is based on the MCS index selected by the node and the packet length of the assigned traffic flow. Also, let $t_i$ denote the starting time of the time slot $T_i$ inside *C*. From the perspective of the genetic algorithm, the initial population *P* will contain a number of schedules $SC_k$, where $k \in \{1, .., K\}$ and $K$ is the initial population size. A single chromosome/schedule $SC_k$ contains the $n(T)$ genes with values $t_i$, where $n()$ is the cardinal number of $T$.

To reduce the artificial packet collision and contention between nodes, $t_i$ offsets in the communication cycle, *C*, should be determined. As such, the following optimization problem is defined:

- **Decision variables**: $t_1, \ldots, t_n$ representing the starting times of each time slot inside the communication cycle.
- **Objective function**: minimize the total overlap between all *n* time slots. The total overlap of a given schedule is defined as the sum of overlaps between each pair of slots:

$$O(SC_k) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} OL(T_i, T_j) \tag{1}$$

where,

$$OL(T_i, T_j) = max\Big(0, min\Big(t_i{+}length(T_i), t_j{+}length(T_j)\Big) - max(t_i, t_j)\Big)$$

- **Constrains**:
  - Time slots within communication cycle boundaries: $t_i + length(T_i) \leq C \; for \; i = 1, 2, \ldots, n$
  - Non-negativity constrain: $t_i \leq 0 \; for \; i = 1, 2, \ldots, n$

When a time slot does not need to be shared, a penalty function can be added to the objective function. As such, the objective function value will be increased based on how much the shared time-space deviates for a certain time slot from a given threshold. In that case, the combined objective function, including both the fitness and penalty components, will be:

$$\mathbf{O}(SC_k) = O(SC_k) + P(SC_k) \tag{2}$$

where penalty function is defined as:

$$P(SC_k) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \Big(max\Big(0, OL(T_i, T_j) - h\Big)\Big) \tag{3}$$

where, *h*, is a threshold that can be adjusted according to the problem requirements. If the overlap between two given time slots is below *h*, the penalty is zero.

The above optimization function can be translated to a linear fitness function for the genetic algorithm as in equation 4:

$$F_1 = \frac{1}{|\sum_{i=1}^{n} T_i - C - \mathbf{O}(SC_k)|} \tag{4}$$

for $\sum_{i=1}^{n} T_i - C - \mathbf{O}(SC_k) \neq 0$, otherwise fitness function will be 1. This fitness value will check how far the total shared time-space of the actual schedule is from the optimal share space $\sum_{i=1}^{n} T_i - C$. By writing it as a ratio we bound the fitness function values between 0 and 1. As such, maximizing the fitness function (that is the aim of the genetic algorithm) will minimize the objective function. The fitness function reflects the quality of the solution based on the total overlap, and the penalty function penalizes schedules for violating the non-overlap constraint for certain time slots. The algorithm need to strike a balance between minimizing overlap (maximizing fitness) and avoiding excessive penalties. Additional penalties will always decrease the solution's fitness.

### *C. Optimization for dedicated time slots*

Even when multiple LTS and BE traffic flows share the same airtime, it would be advantageous to allocate certain time slots that do not overlap with others, as outlined in Section IV-A. This ensures some packets are transmitted without contention at the beginning of time slots. While solutions may share the same airtime cycle, the number of time slots available for

interference-free transmission may differ. To optimize this, we define the following objective function:

$$CNT\_TS = cnt(T_i)\ such\ that\ T_i \notin OL\{t_j > t_i + DIFS\} \quad for\ i, j \leq n\ and\ i \neq j \quad (5)$$

Here, $cnt()$ denotes a count function, and $OL()$ represents a subset of the time slot set, $T$, comprising slots overlapped by others after a DIFS from their starting time, $t_i$. By maximizing the count of non-overlapping time slots, $max(CNT\_TS)$, we enhance the W-TSN schedule.

Based on the objective function, the fitness function will be:

$$F_2 = \frac{CNT\_TS}{n(T)} \quad (6)$$

where $n()$ represent the cardinal number of set $T$. The fitness function values will always be between 0 and 1 and represent how far the current schedule is from having all their time slots not overlapped at their $t_i + DIFS$.

## V. Simulation Environment

This section describes the simulation environment and the tools used. The following subsections present results from various scenarios emulating different amounts of shared air time. We highlight the most significant results from the diverse combinations of methods used in the genetic algorithms to provide a comprehensive understanding of the outcomes.

### A. PYGad Library

The PyGAD library [14] in Python offers a robust toolkit for using genetic algorithms across various scenarios. Users can configure each step, including the initial generation size, parent selection, crossover and mutation types, as well as survivor selection methods. Additionally, parameters like the percentage of genes to mutate and the number of tournaments for parent selection can be customized. The most important aspect is the user-defined fitness function.

### B. Simulation environment

During the simulation, we identify three main factors affecting the quality of the schedule generated by the genetic algorithm: (i) the initial population's impact on finding the optimal W-TSN schedule, (ii) the influence of various method and parameter combinations at different genetic algorithm steps on the final fitness value, and (iii) the effect of two fitness functions defined in section IV.

Diversity in the initial population improves the chances of finding an optimal solution and should be maintained in each generation. However, in the case of W-TSN scheduling, heuristic formation of schedules might impact the optimality of the solution. As such, we specified two initial population options: one fully randomized and one mixed (50% random, 50% heuristic). Heuristics encompass various strategies to generate a W-TSN schedule. The first strategy involves arranging time slots sequentially within the communication cycle until all slots are assigned. If the total airtime of the time slots exceeds the length of the communication cycle, slots will cycle back to the beginning once the cycle length is reached. The second strategy is to allocate all time slots at the start of the communication cycle. Other strategies include ordering the time slots from the largest one to the smallest one and then utilizing the first assigning strategy. The last used strategy orders the time slots from the smallest one to the largest one and then uses the first assigning strategy. In addition to this the size of population should be kept controlled, so does not impact on the fitness calculation speed. In our case, we limited the initial population size to 10 chromosomes, with the same number of genes per chromosome.

To determine which method and which parameter selection on each genetic algorithm step will perform the best in our case, we specified all possible combinations between them. The following parent selection algorithms [”tournament”, ”rank”, ”random”, ”steady-state (sss)”] are tested. For tournament selection, a random sample of chromosomes, *k*, is set at 30% of the population size. The number of parents mating was varied from the following set [2, 4, 6, 8, 10]. For crossover methods, we utilize single- and double-point crossover, scattered crossover, and uniform crossover. The tested mutation methods include: random mutation, swapping mutation, inversion mutation, scramble mutation, and adaptive mutation. In terms of the percentage of genes to be mutated we tested for the following combination set [10, 20, 30, 40, 50]. So the number of total combinations of different methods and parameters in each genetic algorithm step was 2000 in total. Adding here two different initial population types, the total combinations that were tested for each use case was 4000.

For testing the impact of fitness function on achieved optimality of the final W-TSN schedule we specified several use cases depending on the percentage exceeding overall time slot lengths compared to communication cycle length for LTS and BE traffic. The tested exceed values were set at 5%, 10%, 20%, 40%, and 50%. For each use case, we explored all the possible combinations mentioned in the previous paragraph. By exploring these different scenarios, the algorithm can effectively address a wide range of optimization objectives and problem constraints. Moreover, we can conclude which methods are most optimal to be chosen on each genetic algorithm step.

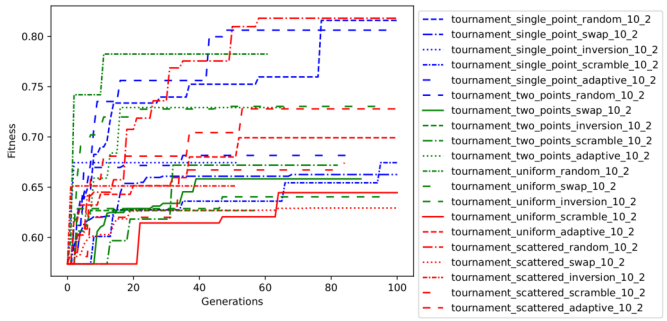
Simulations are performed as follows. For each use case, the initial population is generated and then all the combinations are tested for 100 generation rounds.

## VI. Results

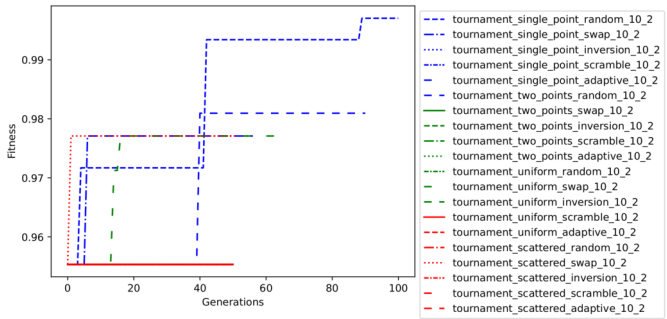
The following subsections present the fitness values analyzing different variations.

### A. Impact of initial population

For both initial population types (fully random and random mixed with heuristics), we compared various combinations of methods and parameters in each step of the genetic algorithms. Figure 3 shows fitness value updates over generations for a sub-set of cases when tournament-based parent selection was




(a) Fully random initial population




(b) Mixed 50% random population with 50% heuristics

Fig. 3: Fitness values update over different generations for different combinations of tournament-based parent selection. $F_1$ function was used. NOTE that the legend shows the following $parent\_selection\ cross-over\ mutation\ mutation\_\%$ and $parents\_mating$

used combined with different methods for cross-over and mutation. The mutation percentage and number of parents mating was kept fixed at 10% and 2, respectively. Here, the total time slot lengths exceed the cycle length by 40%. With a fully randomized population, the initial fitness value is low at 0.5 regardless of the method and parameter combination (Figure 3a). Conversely, a mixed initial population with 50% heuristics achieves an initial fitness value above 0.95 (Figure 3b). These graphs indicate that heuristic-based initial populations lead to optimal W-TSN schedules after several generations, with a fitness value exceeding 0.99. Moreover, even without applying genetic algorithms, heuristic solutions provide better initial W-TSN schedules than fully randomized populations.

### *B. Impact of the selection of different parameters on the overall performance*

Initial tests showed that different combinations of methods and parameters yield different fitness values for the same initial population. To identify the combination that gives the highest fitness ($F_1$) value, we tested 2000 different combinations as specified in subsection V-B.

Table I shows the best combination for each use case, determined by the percentage that total air time of time slots exceeds the communication cycle. When the excess was small (5% and 10%), the fitness value after 100 generations remained low for fully randomized initial populations. This is because small excesses allow most time slots to be scheduled sequentially, reducing shared air time. However, random assignments result in higher shared air time initially, which can not be fixed even after 100 generations. A similar effect was observed for 20% excess, with a fitness of 0.6. In contrast, mixed initial populations consistently achieved fitness values above 0.99.

The best and most frequent method for parent selection was tournament-based selection. Single-point crossover was found to produce the highest fitness values, likely due to our chromosome length of 45-50 genes; longer lengths may require further testing. For mutations, random mutation was most effective, with 10% being the optimal percentage of mutative genes.

### *C. Impact of fitness function*

The fitness value varies based on the optimization function. For $F_2$ (defined in section IV-C), the genetic algorithm aims to maximize the number of interference-free time slots at their start. Optimizing for interference-free slots often results in a schedule that is far from the target of having all slots interference-free at their start.

Figure 4 shows the fitness values over different generations for sub-set of cases when tournament-based parent selection was used combined with different methods for cross-over and mutation. The mutation percentage and number of parents mating was kept fixed at 10% and 2, respectively. The excess in total time slot air time over the communication cycle was set at 50%. The fitness values are much lower compared to Figure 3 due to the more challenging optimization target. Achieving all interference-free slots at their start is harder with a larger airtime excess. However, the relationship between fitness value and initial population remains consistent: mixed populations (random and heuristic) result in higher fitness values.

## VII. Conclusion and Future Work

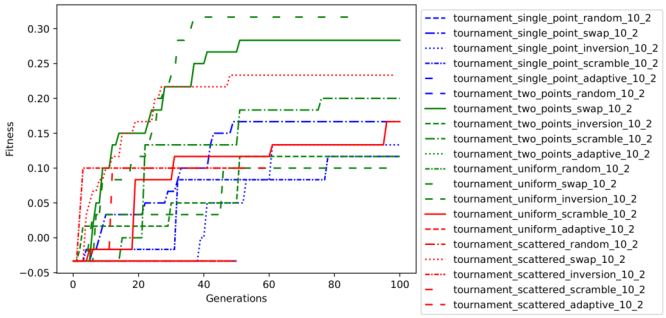
In this paper, we showed how genetic algorithms can be utilized for generating W-TSN schedules where LTS and BE traffic flows can shared the same air time and network. We defined two different fitness functions. One that optimizes the overall shared time slots in the schedule by minimizing that, and the second one that maximizes the number of time slots in the communication cycle that can be used interference-free.

Utilizing the PyGAD library in Python we simulate all the possible combinations for different selected methods and parameters in different steps of the genetic algorithm. We demonstrate that the tournament-based parent selection method consistently yields the highest fitness values across all cases. Single-point cross-over method proved to be the best choice in combination with the random mutation method.

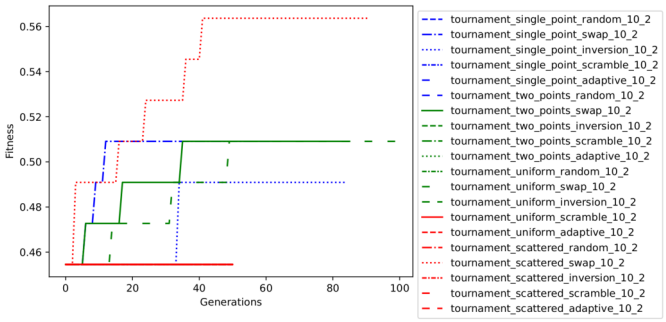
Regarding the impact of the initial population, it was noticed that a fully randomized initial population does not give the highest fitness function even after 100 generations. For this, a mixed initial population where 50% of chromosomes are generated randomly and 50% of them are generated with certain heuristic techniques gives a higher fitness value when optimized for the air-time overlap. In the case of a fitness

| Use case | Initial population | Parent selection | Cross-over | Mutation | Mutation % | Nr. parents mating | Fitness value |
|---|---|---|---|---|---|---|---|
| 5% | Fully random | Tournament | Scattered | Random | 10 | 8 | 0.23 |
| | Mixed | Tournament | Single point | Random | 10 | 2 | 1 |
| 10% | Fully random | SSS | Uniform | Swapping | 30 | 4 | 0.5 |
| | Mixed | SSS | Single point | Swapping | 50 | 10 | 0.99 |
| 20% | Fully random | Tournament | Two points | Random | 10 | 8 | 0.6 |
| | Mixed | Tournament | Single point | Swapping | 50 | 4 | 0.99 |
| 40% | Fully random | Tournament | Single point | Random | 10 | 8 | 0.94 |
| | Mixed | Tournament | Two points | Random | 10 | 8 | 0.99 |
| 50% | Fully random | Tournament | Scattered | Random | 10 | 2 | 0.98 |
| | Mixed | Tournament | Single point | Random | 50 | 4 | 0.99 |

TABLE I: The best combination of methods/parameters for each use case and initial population with fitness value after 100th generation.




(a) Fully random initial population




(b) Mixed 50% random population with 50% heuristics

Fig. 4: Fitness values update over different generations for different combinations of tournament-based parent selection. $F_2$ function was used. NOTE that the legend shows the following $parent\_selection\ cross-over\ mutation\ mutation\_\%$ and $parents\_mating$

function that optimizes the number of interference-free time slots the achieved fitness values were lower due to the hardest target to reach (all time slots being interference-free). Future work will be to test such algorithms in real network scenarios and how to integrate such algorithms with network controller.

## ACKNOWLEDGMENT

This research was partially funded by the Flemish FWO SBO S003921N VERI-END.com project and the Flemish Government under the "Onderzoeksprogramma Artificiële Intelligentie (AI) Vlaanderen" program.

## REFERENCES

[1] A. Larrañaga, M. C. Lucas-Estañ, I. Martinez, I. Val, and J. Gozalvez, "Analysis of 5G-TSN Integration to Support Industry 4.0," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2020-September, pp. 1111–1114, 9 2020.

[2] H. Chahed and A. Kassler, "TSN Network Scheduling—Challenges and Approaches," *Network 2023, Vol. 3, Pages 585-624*, vol. 3, no. 4, pp. 585–624, 12 2023.

[3] L. De Moura and Bjorner Nikolaj, "Satisfiability Modulo Theories: Introduction and Applications," *Communications of the ACM*, vol. 54, no. 9, pp. 69–77, 9 2011.

[4] V. Gavrilut and P. Pop, "Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications," *IEEE WFCS - Proceedings*, vol. 2018-June, pp. 1–4, 7 2018.

[5] F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (TSN)," *ACM International Conference Proceeding Series*, vol. 19-21-October-2016, pp. 203–212, 10 2016.

[6] Y. Wang, P. Yu, and Y. Cheng, "Joint Scheduling Algorithm of Routing and GCL Based on Tabu Search," 2022. [Online]. Available: https://doi.org/10.21203/rs.3.rs-1713179/v1

[7] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and Scheduling of Time-Triggered Traffic in Time-Sensitive Networks," *IEEE Transactions on Industrial Informatics*, no. 7, 7 2020.

[8] X. He, X. Zhuge, F. Dang, W. Xu, and Z. Yang, "DeepScheduler: Enabling Flow-Aware Scheduling in Time-Sensitive Networking," *IEEE INFOCOM 2023*, 2023.

[9] M. Pahlevan and R. Obermaisser, "Genetic Algorithm for Scheduling Time-Triggered Traffic in Time-Sensitive Networks," *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, vol. 2018-September, pp. 337–344, 10 2018.

[10] C. Liu, Y. Hong, J. Wang, C. Liu Sr, L. Tian, and J. Xu, "Hierarchical cross traffic scheduling based on time-aware shapers for mobile time-sensitive fronthaul network," *Wireless Communications and Mobile Computing*, vol. 2024, no. 1, p. 8882006, 2024.

[11] Z. Li, J. Yang, C. Guo, J. Xiao, T. Tao, and C. Li, "A Joint Scheduling Scheme for WiFi Access TSN," *Sensors 2024, Vol. 24, Page 2554*, vol. 24, no. 8, p. 2554, 4 2024.

[12] M. Michell, "An introduction to genetic algorithms," 1998. [Online]. Available: http://books.google.com/books?id=0eznlz0TF-IC

[13] Özkaya, J. Haxhibeqiri, I. Moerman, and J. Hoebeke, "Simulating and validating openwifi W-TSN in ns-3," *IEEE WFCS2024, the 20th IEEE International Conference on Factory Communication Systems*, 2024.

[14] A. F. Gad, "PyGAD: An Intuitive Genetic Algorithm Python Library," *Multimedia Tools and Applications*, 6 2021.