Evaluation of Metaheuristic Algorithms for TAS Scheduling in Time-Sensitive Networking

Junhong Min, MyoungJin Oh, Woongsoo Kim, Hyewon Seo, and Jeongyeup Paek
Department of Computer Science & Engineering
Chung-Ang University
Seoul, Republic of Korea
{dmc93, omjin7n, woongsu0614, hyewon981019, jpaek}@cau.ac.kr

***Abstract*—Time Sensitive Networking (TSN) is an emerging technology for providing deterministic ultra low-latency network based on Ethernet. It is designed for application domains such as industrial automation where guaranteed latency is required to meet the hard deadlines of flows. TSN achieves this by carefully scheduling the transmissions of frames through the IEEE 802.1Qbv standard, also known as time-aware shaper (TAS). However, TAS scheduling problem is classified as NP-hard. To overcome this challenge, we explore various metaheuristic approaches using MEALPY, a state-of-the-art meta-heuristic algorithm module of Python. We evaluate the meta-heuristic algorithms for TAS scheduling optimization problem through TSN simulations and provide observations for future directions.**

***Index Terms*—IEEE 802.1Q, IEEE 802.1Qbv, Time-Sensitive Network (TSN), Metaheuristic, MEALPY, Flow scheduling**

I. Introduction

Ethernet is one of the most widely used network technologies in various fields. With the introduction of optical fiber cables, today's Ethernet can provide datarates of 100 Gbps. However, due to its best-effort service characteristics, Ethernet has no guarantee in latency despite the high datarate. Therefore, proprietary Ethernet variants such as Profinet, EtherCAT, or fieldbus technologies such as Profibus DP or Modbus-RTU are used in industrial fields that require deterministic communication with guaranteed arrival time. The problem is, these technologies are incompatible with each other, not even with the standard Ethernet, leading to multiple segmented networks, limitations on supported devices, customer lock-in, and eventually higher cost.

To address this problem, IEEE 802.1 working group [1] launched Time-Sensitive Networking (TSN) task group [2] in 2012 to unify various proprietary Ethernet-based technologies. TSN intends to provide deterministic ultra low-latency, low-jitter, and zero congestion loss network required in industrial domains such as factory automation, in-vehicle, and avionic networks. With this goal, TSN standardized *time-aware shaper* (TAS) in IEEE 802.1Qbv [3] (with a few other standards) to achieve deterministic low-latency via flow scheduling. However, these standards present only the necessity and outlines of scheduling. Algorithm for actual flow scheduling is neither defined nor suggested. Thus, TAS scheduling has emerged as a major research topic in TSN.

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (2022R1A4A5034130 & 2021R1A2C1008840), and also by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2022-RS-2022-00156353) supervised by the IITP (Institute for Information & Communications Technology Planning & Evaluation)

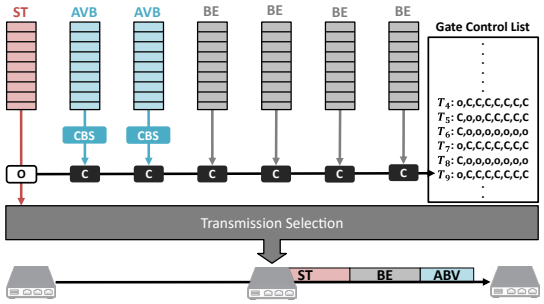

Fig. 1. IEEE 802.1Qbv Time Aware Shaping

Flow scheduling in TSN can be regarded as a *flow-shop problem*. By default, TSN assumes that flows with specific latency requirements are transmitted periodically, and provide mechanisms considering the cycles of all flows to satisfy latency requirements. As such, several prior works in the literature suggest a constraint-programming (CP) approach. CP defines the requirements of flows as constraints, and in general, the number of flows and constraints are proportional. However, computing resources and execution time required by CP solvers grow exponentially rather than linear to the number of flows. Therefore, the need for metaheuristic approaches that provide reasonable running times is emerging.

In this work, we present *packet size step scheduling* (PSSS) algorithm for no-wait scheduling of TAS flows, explore scheduling optimizations through various metaheuristic approaches to solve the flow scheduling problem of TSN, and evaluate the proposals using Meaply [4], a state-of-the-art metaheuristic algorithm module in Python. Our work focuses on the potentials of various metaheuristic approaches for TSN flow scheduling optimization problem. According to our result, metaheuristic algorithms from system-based and math-based ideas generally have good performance. Artificial ecosystem-based optimization [5] (system-based) achieves a performance gain of up to 49% compared to other algorithms.

II. Related Work

Several prior works proposed CP-based methods for NP-hard TAS scheduling problem, and found it difficult to provide practical execution time for relatively large network topology and flow sets [6]–[9]. According to the comprehensive TSN survey by Seol *et al.* [10], metaheuristic-based algorithms are proposed as practical alternatives to address TSN problems. Macchiaroli *et al.* [11] proposed heuristic-based no-wait flow scheduling which performs optimization through Tabu search algorithm, and it is extended to no-wait TAS scheduling for TSN by Dürr *et al.* [12]. Hellmanns *et al.* [8] proposed a two-stage approach including a heuristic Tabu search algorithm, and Reusch *et al.* [9] proposed a simulated annealing-based algorithm to find a practical solution. However, these approaches are limited to only a few metaheuristic algorithms. We explore a variety of alternatives to seek for enhancements.

III. System model

In this section, we propose *packet size step scheduling (PSSS)* algorithm for TSN flow scheduling, and consider various metaheuristic optimization strategies to improve scheduling performance. We focus on no-wait flow scheduling [11], [12] to minimize flow latency and prevent packet drops due to buffer overflow.

A. TAS and Gate Control List

Fig. 1 illustrates how a switch handles scheduled flows according to IEEE 802.1Qbv. Traffic classes are generally categorized into *audio video bridging (AVB)*, *scheduled time-critical (ST)* (a.k.a. Time-Triggered (TT)), and *best-effort (BE)* traffic. Transmission schedule for each class is stored in the form of *gate control list (GCL)* which contains port state information for each traffic class queue, either *Close(c)* or *Open(o)*. Transmission is permitted only when a queue is in open state. ST traffic usually has a strict latency deadline, and the TSN schedule expressed in GCL must satisfy this requirement.

TSN considers periodic transmissions for time-critical flows; i.e. all ST flows that require scheduling are assumed to have periodicity. Thus it is possible to periodically repeat a GCL schedule reflecting the periods of all ST traffic. A cycle in which the schedule is repeated is called a *hyper-period*, and the least common multiple of the cycles of all flows is generally set as the hyper-period. Transmission plans of various traffic types are included in the GCL schedule for the hyper-period, which is reflected as information on port gate state control at specific time T_n as shown in GCL in Fig. 1.

B. Flow scheduling

In the flow scheduling model of TSN, each flow must be scheduled avoiding the transmission schedule of other flows. Thus, time space within each hyper-period is limited. For lower-latency, it is always preferred to transmit a flow at the beginning of a hyper-period. But in reality, interference between flows must be considered. Therefore, Dürr's algorithm *et al.* [12] performs scheduling at the earliest possible time. When a schedule collision occurs, the start time is moved by the size of the collision, and the scheduling is repeated. This method may cause frequent re-scheduling operations if there are many packets scheduled previously, and can be a burden in an environment where scheduling is re-tried a large number of times (optimization process through repetitive scheduling). Therefore, we introduce PSSS algorithm to reduce the number of searches that find appropriate scheduling start time.

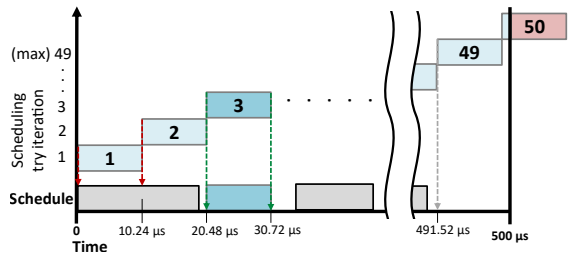

Fig. 2. Example operation of PSSS

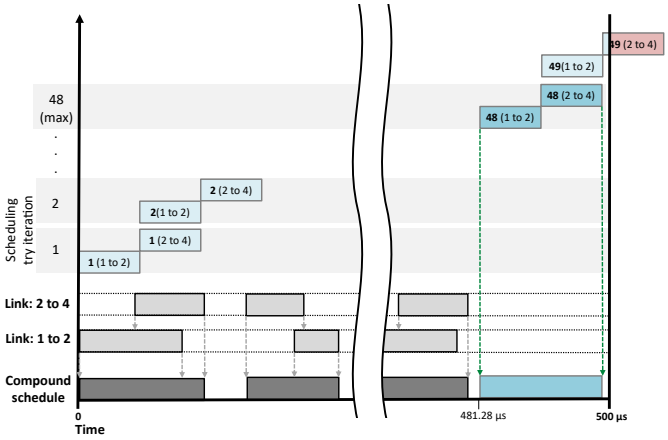

Fig. 3. Example operation of PSSS with multi hops

Fig. 2 depicts the scheduling operation of PSSS. PSSS defers the start position as much as the transmission delay of the packets to be scheduled at that start position when scheduling is failed. When packet size is 128 bytes and link bandwidth is 100 Mbps, transmission delay is 10.24 μs. If the start position of the first scheduling attempt is 0 μs, start position is changed to 0 μs$\rightarrow$10.24 μs$\rightarrow$20.48 μs whenever the scheduling attempt fails. Our scheme assumes that no transmission schedule can go beyond the hyper-period. Therefore, if the hyper-period of GCL is 500 μs, the maximum theoretical scheduling attempt of a flow is 500/10.24. However, if the transmission path of a flow is multihop, the number of actual meaningful scheduling retries decreases as shown in Fig. 3. In addition, if there is an additional scheduling time constraint according to the transmission interval, the number of meaningful retries is further reduced as shown in Fig. 4. Therefore, the maximum number of flow scheduling Num_{sch} is:

$$Num_{sch} = \lfloor Interval\ Period \div Tx\ Delay \rfloor - Path\ length + 1 \quad (1)$$

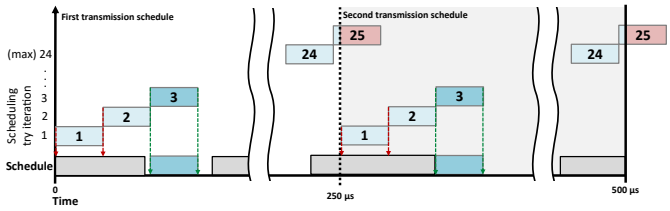

Fig. 4. Example operation of PSSS with transmission interval

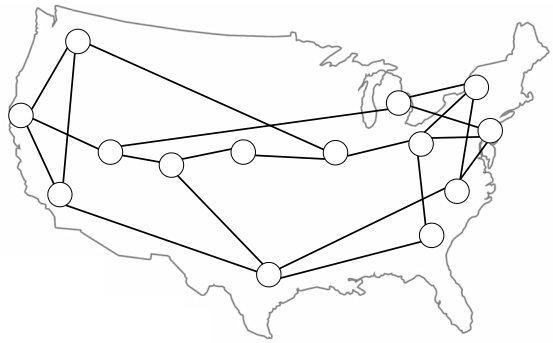

Fig. 5. NSF network topology

PSSS can reduce the potential maximum number of scheduling tries by searching for a scheduling start position based on their packet size.

C. Scheduling optimization

In heuristic flow scheduling, the performance may vary depending on the order in which the flows are scheduled. Scheduling order of flows can be expressed as a permutation, and the optimization of flow scheduling is to find a scheduling permutation (order) that allows as many flows to comply with the transmission deadline requirement. This is one of combinatorial optimization problems that are considered as NP-hard, so metaheuristic-based approaches can be a practical strategy. To this end, we introduce metaheuristic algorithms that we intend to apply to TSN flow scheduling.

We first define the cost function and permutation to optimize, and then perform optimization through various metaheuristic algorithms provided by Mealpy [4]. In our scheme, the cost function is the number of flows that failed to be scheduled, and the permutation is the order of flows that is scheduled by PSSS. The lower the cost, the closer the solution is to the optimum. Our work considers 28 metaheuristic algorithms in seven categories on Table I, and this category classification is based on Mealpy's documentation [4] and related literature [13].

IV. Evaluation

We evaluate the optimization performance of metaheuristic algorithms in Table I. Each metaheuristic algorithm considers the scheduling result of PSSS as their cost, and performs flow scheduling order optimization to reduce the cost.

TABLE I
METAHEURISTIC ALGORITHMS IN MEALPY

Algorithm type	Algorithm name
Evolutionary	Evolutionary Programming (BaseEP) Evolution Strategies (BaseES) Memetic Algorithm (BaseMA) Genetic Algorithm (BaseGA)
Swarm	Particle Swarm Optimization (BasePSO) Bees Algorithm (BaseBeesA) Ant Colony Optimization (BaseACOR) Whale Optimization Algorithm (BaseWOA)
Physics	Simulated Annealling (BaseSA) Electromagnetic Field Opt. (OriginalEFO) Atom Search Optimization (BaseASO) Equilibrium Optimizer (BaseEO)
Human	Culture Algorithm (OriginalCA) Teaching Learning-based Opt. (OriginalTLO) Brain Storm Optimization (BaseBSO) Queuing Search Algorithm (OriginalQSA)
Bio	Invasive Weed Optimization (OriginalIWO) Biogeography-Based Opt. (OriginalBBO)
System	Germinal Center Optimization (OriginalGCO) Artificial Ecosystem-based Opt. (OriginalAEO)
Math	Hill Climbing (OriginalHC) Sine Cosine Algorithm (OriginalSCA) Gradient-Based Optimizer (OriginalGBO) Arithmetic Optimization Algorithm (OrginalAOA)

TABLE II
FLOWS SPECIFICATION

Flow type	Size	Interval	deadline	# of flows
ST1	64 bytes	0.2 ms	0.2 ms	100
ST2	128 bytes	0.5 ms	0.5 ms	100
ST3	196 bytes	1 ms	1 ms	100

A. Simulation setup

In our TSN scenario, the National Science Foundation (NSF) network in Fig. 5 is used as the network topology and TSN flows follow the specification in Table II. All links are full-duplex with 100 Mbps bandwidth. We conduct simulations for 30 minutes for each optimization algorithm on a workstation with Intel Core i7-8700 and 16GB memory. Our simulator is implemented in Python 3.7, and we use Mealpy 2.4.2 for metaheuristic algorithms. For practical runtime (within 30 minutes), we set a population size *(pop_size)* to 10 as a default for each metaheuristic algorithm. Other parameter settings follow the default settings of Mealpy except when they cause errors. In those cases (BaseMA, BaseACOR, BaseBSO, OriginalWO, OriginalHC), we set *pop_size* to 50.

B. Simulation results

Fig. 6 shows the best cost of each algorithm. The best cost means the lowest cost obtained from the optimization process. According to Fig. 6, system-based and math-based algorithms show uniformly good optimization performance. Although our simulation does not represent all algorithms in each category, we can expect that it would be relatively good to use the system and math-based metaheuristic algorithms for TSN scheduling optimization. In particular, Artificial Ecosystem-based Optimization (OriginalAEO) achieves a lower cost of up to 49% compared to other algorithms.

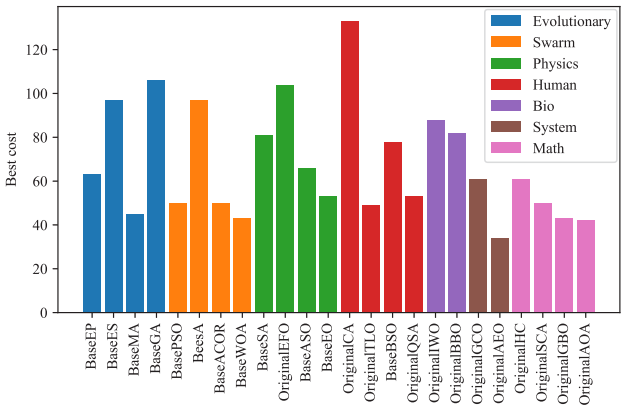

Fig. 6. Best cost of each algorithm

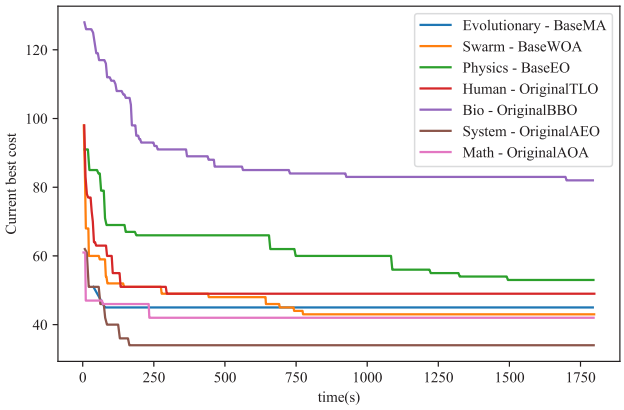

Fig. 7. Best cost over execution times

Fig. 7 shows the best cost changes over time for the seven algorithms that showed the best results in each category. This result shows that OriginalAEO returns a better solution (lower cost) than all other metaheuristic algorithms in a short execution time of 163.9 seconds. It is the second fastest convergence speed among the seven metaheuristics in Fig. 7. This means that OriginalAEO is superior not only in solution quality but also in optimization speed. Also, math-based OriginalAoA shows relatively superior convergence speed compared to other algorithms.

Our results show the potentials of each metaheuristic approach in TSN schedule optimization scenarios, and strongly implies that Artificial Ecosystem-based Optimization can be a plausible strategy.

V. Conclusion

We presented *packet size step scheduling (PSSS)* algorithm for no-wait TAS scheduling in TSN, and explored and evaluated various metaheuristic algorithms as an optimization method. Our evaluation results provide observations on the direction of the metaheuristic algorithms for TSN flow scheduling. As our future work, we intend to design an algorithm for solving joint routing and TAS scheduling problem using metaheuristic algorithms based on the findings from this work.

References

[1] "IEEE 802.1 Working Group," 2021, [last accessed Jul. 2022]. [Online]. Available: https://1.ieee802.org/

[2] "IEEE Time-Sensitive Networking Task Group," 2017, [last accessed Jul. 2022]. [Online]. Available: http://www.ieee802.org/1/pages/tsn.html

[3] *IEEE Standard for Local and Metropolitan Area Networks – Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic*, IEEE Std. 802.1Qbv-2015, pp. 1-57, 2016.

[4] N. V. Thieu, "A collection of the state-of-the-art MEta-heuristics ALgorithms in PYthon: Mealpy," 2020, [last accessed Jul. 2022]. [Online]. Available: https://doi.org/10.5281/zenodo.3711948

[5] W. Zhao, L. Wang, and Z. Zhang, "Artificial ecosystem-based optimization: a novel nature-inspired meta-heuristic algorithm," *Neural Computing and Applications*, vol. 32, no. 13, pp. 9383–9425, 2020.

[6] J. Falk, F. Dürr, and K. Rothermel, "Exploring practical limitations of joint routing and scheduling for TSN with ILP," in *2018 IEEE 24th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, pp. 136–146.

[7] W. Steiner, S. S. Craciunas, and R. S. Oliver, "Traffic planning for time-sensitive communication," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 42–47, 2018.

[8] D. Hellmanns, A. Glavackij, J. Falk, R. Hummen, S. Kehrer, and F. Dürr, "Scaling TSN scheduling for factory automation networks," in *2020 16th IEEE International Conference on Factory Communication Systems (WFCS)*. IEEE, pp. 1–8.

[9] N. Reusch, S. S. Craciunas, and P. Pop, "Dependability-aware routing and scheduling for Time-Sensitive Networking," *IET Cyber-Physical Systems: Theory & Applications*, 2022.

[10] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely Survey of Time-Sensitive Networking: Past and Future Directions," *IEEE Access*, vol. 9, pp. 142 506–142 527, 2021.

[11] R. Macchiaroli, S. Mole, and S. Riemma, "Modelling and optimization of industrial manufacturing processes subject to no-wait constraints," *International Journal of Production Research*, vol. 37, no. 11, pp. 2585–2607, 1999.

[12] "No-wait packet scheduling for IEEE time-sensitive networks (TSN), author=Dürr, Frank and Nayak, Naresh Ganesh," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, 2016, pp. 203–212.

[13] T. Nguyen, G. Nguyen, and B. M. Nguyen, "EO-CNN: an enhanced CNN model trained by equilibrium optimization for traffic transportation prediction," *Procedia Computer Science*, vol. 176, pp. 800–809, 2020.