Optimal Graph Partitioning for Time-Sensitive Flow Scheduling towards Digital Twin Networks

Shuo Wang
Swinburne University of Technology
Melbourne, VIC, Australia

Jonathan Kua
Deakin University
Geelong, VIC, Australia

Jiong Jin
Swinburne University of Technology
Melbourne, VIC, Australia

Ambarish Kulkarni
Swinburne University of Technology
Melbourne, VIC, Australia

Prem Prakash Jayaraman
Swinburne University of Technology
Melbourne, VIC, Australia

Xianghui Cao
Southeast University
Nanjing, Jiangsu, China

ABSTRACT

The growing maturity of Digital Twin (DT) technology represents a quantum leap towards the realisation of Industry 4.0 and beyond. DT refers to virtual representations (in a virtual space) of physical objects/processes/systems (in a physical space), where information is regularly exchanged between them to enable real-time remote control and monitoring. DT will significantly improve product life-cycles and will transform industries such as smart manufacturing, smart transportation, and so forth. Digital Twin Networks (DTNs) are envisaged to be the norm where multiple DTs are logically connected to their respective physical objects, forming a many-to-many communication relationship. Strict real-time communication for bi-directional data flows is required in DTNs for DTs to accurately reflect the changes in the physical objects, and vice-versa. One potential candidate to achieve real-time data transmission is Time Sensitive Networking (TSN). The IEEE 802.1Q working group has developed a set of TSN standards to facilitate data transmissions that require low-latency, high availability and reliability. In TSN, the Central Network Controller (CNC) computes the schedule of frame transmission. However, the computational time required can exponentially increase as the number of nodes and data flows increases (already typical in industrial environments and will increase exponentially with DTNs). In this paper, we propose a novel technique using multi-level graph partitioning theory with Integer Linear Programming (ILP) to facilitate TSN scheduling. Our results demonstrated significant improvements in computational time and the success rate of scheduled data flows in complex networks where there are up to 100 nodes and 350 data flows.

KEYWORDS

Digital Twin Networks, Time Sensitive Networks, Flow Scheduling, Graph Partitioning, Integer Linear Programming

Digital Twin & Edge AI for Industrial IoT '22, October 17, 2022, Sydney, NSW, Australia

ACM ISBN 978-1-4503-9784-1/22/10…$15.00
https://doi.org/10.1145/3566099.3569003

ACM Reference Format:
Shuo Wang, Jonathan Kua, Jiong Jin, Ambarish Kulkarni, Prem Prakash Jayaraman, and Xianghui Cao. 2022. Optimal Graph Partitioning for Time-Sensitive Flow Scheduling towards Digital Twin Networks. In *1st Workshop on Digital Twin & Edge AI for IIoT (Digital Twin & Edge AI for Industrial IoT '22), October 17, 2022, Sydney, NSW, Australia.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3566099.3569003

1 INTRODUCTION

Digital Twin (DT) technology has generated significant amount of interest in recent years, in part propelled by the rapid developments in Artificial Intelligence (AI), Machine Learning (ML), Cloud/Edge computing paradigms and capabilities, along with the convergence of Cyber-physical Systems (CPS) and Industrial Internet of Things (IIoT) technologies towards Industry 4.0 [1, 2].

DT promises to revolutionise industries by creating virtual representations/twins (in a virtual world) of physical objects (in a physical space), where information is regularly exchanged between them to enable real-time modelling, remote control and monitoring of systems and processes [3]. In addition to simulating the states and behaviours of the physical objects, DT also enable physical objects to be controlled from the virtual twins. The maturity of DT technology will significantly improve product life-cycles and reduce produce development costs, hence transforming industries such as smart manufacturing, smart transportation, and so forth.

With the use of DTs gaining traction, Digital Twin Networks (DTNs) are envisaged to be the norm where multiple DTs are logically connected to their respective physical objects across the communication network, forming a many-to-many relationship [4]. One major challenge of achieving fully-functional DTNs is network connectivity and provisioning. DTN requires (ultra)-low latency, high reliability and high bandwidth capacity. The models within the virtual twins need to be regularly updated to accurately reflect the changes in the physical objects, and physical objects need to be able to respond to the feedback provided by the virtual twins in a timely manner. Therefore, strict real-time communication for bi-directional data flows need to be adhered in DTNs. High bandwidth capacity is also required as data volume is expected to increase exponentially with the number of DTs. Currently, it is challenging to achieve this level of network performance in industrial settings, especially when the communications infrastructure is shared with other conventional/non-DT traffic flows.

One potential candidate for meeting the strict real-time requirements of DT flows in industrial environments is Time Sensitive

Networking [5]. The IEEE 802.1Q working group has developed a set of TSN standards to schedule and regulate data transmissions over wired networks for time-sensitive applications, where the requirements to deliver data within a specific time frame need to be met, hence satisfying the defined Quality of Service (QoS) requirements, including the requirements for DT traffic flows [6].

In TSN, industrial control applications/end-nodes communicate their deterministic requirements to the Centralised User Configuration (CUC) [7]. The CUC will then send this information to the Central Network Controller (CNC) which is responsible for learning the physical network topology, aggregating all the requests, computing their routes, and schedules the transmission for each data flow. Upon completion, it will then send the computed schedule to the the respective TSN bridges. The CNC is a software application that runs locally on-premises and it can be hosted onto any computational hardware, ranging from a full-size industrial PC to an embedded device at the edge on the factory floor.

Our work focuses on facilitating TSN scheduling in accordance with the IEEE 802.1Qbv (scheduling and traffic shaping) standard. IEEE 802.1Qbv defines algorithms that allow different network flows within the TSN network to be scheduled and transmitted with a bounded latency with minimum jitter. The scheduler is responsible for defining specific packets transmitted within the designated period. The scheduling problem in TSN is often considered as a bin-packing problem, which has also been proved to be an NP-complete problem. There are several algorithms for solving such problems over the years, such as Satisfiability Modulo Theories (SMT) [8], Integer Linear Programming (ILP) [9] and other heuristics-based approaches [10]. However, it is acknowledged that the computational time of the scheduling process increases exponentially when the size of network grows or when there are many simultaneous time-triggered data flows, thus resulting in the TSN-enabled network failing to meet its deterministic QoS requirements.

Inspired by graph partitioning theory in computer science [11], we seek to design and propose a multi-level graph partitioning based algorithm to facilitate TSN scheduling in the context of DTNs. In this work, we consider a scenario where the TSN's CNC software runs on a resource-constrained computing unit with limited memory and processing capabilities, providing the benefits of edge/fog computing in industrial environments (such as real-time decision making, energy efficient computing). Reduction in CNC's computational time without sacrificing the success rate of scheduled flows is crucial in such context.

In this paper, we present a novel algorithm based on multi-level graph partitioning theory and ILP for TSN scheduling. We first represent the network configuration as a graph, then partition the graph into smaller subsets using the multi-level graph partitioning technique. We then use ILP to represent the network flows as an objective function. The algorithm aims to maximise the number of scheduled time-triggered flows given the specific constraints, by recursively generating the time schedule for each subset. We apply our algorithm to two well-known scheduling algorithms, i.e., Scheduling with Pathsets Routing (S/PR) and Scheduling with Fix-path Routing (S/FR). Our evaluation results and analysis show that our algorithm can efficiently divide very complex network graphs (representing complex network configurations) into balanced subsets, significantly reduce the computational time of the scheduling process, and improve the success rate of scheduled flows that meet their latency requirements (where there are up to 100 nodes and 350 data flows).

In summary, we make the following contributions in this paper:

- Design and propose a novel technique that combines multi-level graph partitioning theory and ILP to facilitate the computational of TSN scheduling in the context of DTNs.
- Apply our technique across a wide range of network parameters, and design an optimal graph partitioning algorithm to work with S/PR and S/FR scheduling algorithms.
- Demonstrate significant reduction in computational time and improvements in success rate in complex networks.

The remaining of the paper is structured as follows. Section 2 presents background information and related work, Section 3 introduces our proposed algorithm and Section 4 presents our evaluation results and analysis. Section 5 concludes and outlines future work.

2 BACKGROUND AND RELATED WORK

In this section, we present background information and related work on real-time communication in DTNs, multi-level graph partitioning, and the S/PR and S/FR scheduling techniques in TSN.

2.1 Real-time communications in DTNs

A successful implementation of DTNs require accurate mappings between the physical objects and virtual twins. Real-time requirement needs to be achieved to enable the co-evolution, cooperation and collaboration between these physical and virtual nodes. A full realisation of DT technology involves a many-to-many communication relationship between the nodes. Any changes in the physical objects should be accurately communicated and reflected in the virtual twins within a specified time frame.

Advanced communication technologies are required for Physical-to-Physical (P2P), Physical-to-Virtual (P2V), and Virtual-to-Virtual (V2V) communication [4]. P2P communications are already a commonplace in industrial environments, where various end-points (e.g., devices, sensors, actuators, RFID tags, etc) are interconnected via wired/wireless technologies (e.g., Ethernet variants, TSN, LP-WANs, NB-IoT, LoRa, etc) [12]. P2V and V2V communications are critical for a fully functional DTN. In P2V, the physical object needs to continuously sends/receive data to/from the virtual twin (updating model, receiving feedback, predicting changes, etc) [13, 14]. This is also communicated over various wired/wireless channels.

Since the communication infrastructure is shared with other non-DT data traffic, it is imperative to ensure that the latency and jitter of DT traffic is low, reliability is high (for accurate modelling), network bandwidth is sufficient to accommodate a large number of data flows (DT and other flows sharing the same network). In V2V, data exchanges occur virtually to mimic the physical objects' behaviour. It theoretically has no latency and reliability constraints. The data exchange speed depends on the servers' computing capacities [4].

P2P and P2V typically have hard real-time constraints, hence time-sensitive flow scheduling is required to facilitate their communications. V2V has lower real-time requirements as virtual twins can share data over a relatively longer time period. In our work, we focus on using TSN as an enabler for DTNs, where conventional P2P data flows, share the network with DT's P2P, P2V data flows as

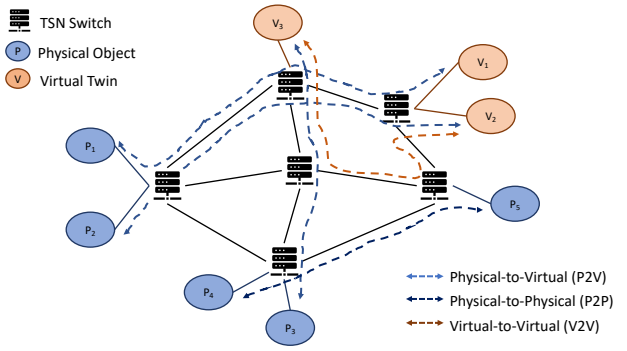


Figure 1: An illustration of Physical-to-Virtual (P2V), Physical-to-Physical (P2P) and Virtual-to-Virtual (V2V) communication in a TSN-enabled DTN. The shortest path is not always the most optimal. Our proposed algorithm considers all (non-DT/DT) flow information across the entire network.

illustrated in Fig. 1. It is important to note that the shortest path is not always the most optimal path. Our proposed algorithm considers the flow information of all flows (including both non-DT and DT data flows) across the entire network.

2.2 Multi-level graph partitioning

In the field of high performance computing, ensuring a good partition of graphs is valuable in managing computational resources. Vertices in the graph are used to represent computational units and the edges are used to represent communication between connected vertices. Multi-level graph partitioning is an enhancement to help manage complex graphs.

A generalisation of an ILP formulation for balanced bi-partitioning is introduced in [15] to solve the general graph partitioning problem. First, binary decision variables for all edges (E) and vertices (V) of the graph are introduced. For each edge the variable $E_{u,v} = \{0, 1\}$, $E_{u,v} = 1$ if a single edge (e) is an edge between node u and v to be cut. For each single vertex (v) in V and partition k, $w_{u,v}$ is used to represent the weight on the edge cut. Constraints are defined to ensure that the result is a balanced and valid k-partition. However, the ILP based partitioning is based on bi-partitioning algorithm, so the optimal case only considered that $k = 2^N$, $N \in \mathbb{Z}^+$. We then use the direct k-way cut function in the METIS to calculate the rest of the minimum edge cut with a less balanced partitioning result. The purpose is to develop an optimal minimum edge cut algorithm using the ILP and direct k-way partitioning.

The balanced graph partitioning is defined as creating k disjoint sets of vertices with the constraint that the sum of the weights in any given set does not exceed corresponding average imbalanced parameter which calculated based on 10 consecutive random partitioning.

$$\text{minimize} \sum_{u,v \in T} w_{u,v} * e_{u,v} \tag{1}$$

2.3 S/PR and S/FR scheduling in TSN

In this work, we have adopted the ILP formulations of two well-known joint routing and scheduling approaches, namely, Scheduling with Pathsets Routing (S/PR) and Scheduling with Fix-path Routing (S/FR) [9]. We use these scheduling schemes as a measurement benchmark for our work. These two approaches aim to maximise the number of allocated time-triggered data flows timeslots on a given network link. We briefly describe their operations and limitations below.

2.3.1 Scheduling with Pathsets Routing (S/PR). The objective of S/PR is to maximise the number of paths with assigned timeslots (as described in Formula 2). A set of candidate paths between the source and destination is being generated. The routes of all time-triggered flows are specified, and the candidate paths of each flow are constructed based on all shortest paths algorithm. In this approach, the search space of routing is limited within the generated set of shortest paths for a corresponding flow.

This scheme will route the flow through one of the paths in the generated set instead of searching the complete graph links for arbitrary paths. This approach has a lower computational time as the time required to compute and generate all shortest paths are negligible compared to the execution time of the ILP solver. The ILP formulation of S/PR is defined as follows:

$$\text{maximize} \sum_{k \in T} \sum_{l \in P} pt_{l,k} \tag{2}$$

where $PT = \{pt_{l,k}\}$ is the ILP variable which maps timeslots to paths. When path l is allocated time slot k, $pt_{l,k} = 1$. Otherwise, $pt_{l,k} = 0$. In this formulation, P is a set of possible paths for a flow, and l indicates the index of the path in this set. T is a set of timeslots over a period, and k is the index of a timeslot.

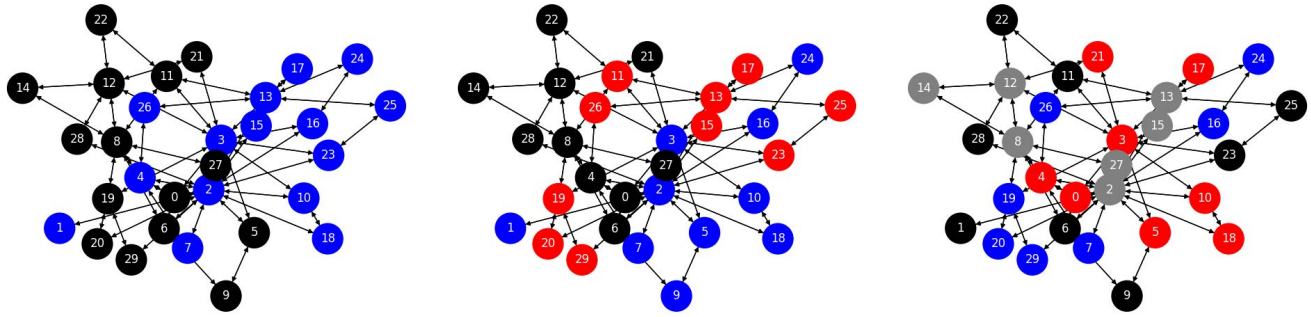
2.3.2 Scheduling with Fix-path Routing (S/FR). The objective of S/FR is to maximise the number of flows to be allocated to available timeslots (as described in Formula 3). This approach extends S/PR by randomly selecting one of the shortest paths of a flow as the assigned path of this flow to further reduce the computational time of finding the optimal schedule. The ILP formulations only define the timeslot allocation for these flows. It is defined as follows:

$$\text{maximize} \sum_{i \in ST} \sum_{k \in T} t_{i,k} \tag{3}$$

where $ST = \{t_{i,k}\}$ is the ILP variables which maps flow to timeslot. When flow i is allocated to time slot k, $t_{i,k} = 1$. Otherwise, $t_{i,k} = 0$. In this formulation, ST is the set of time-triggered flows, and i indicates the index of the flow in this set. T is a set of timeslots over a period, and k is the index of a timeslot.

Both S/PR and S/FR scheduling approaches are defined to avoid the time-triggered flows interfering with each other. They achieve this by adhering to two constraints: (i) allocating one flow to a timeslot in a given link, and (ii) avoiding allocating the same timeslot for more than one flow.

Instead of allocating data flows to specific timeslots, the S/PR approach allocate paths to timeslots, so an additional constraint is imposed, where the usage of the path can only be allocated to



(a) $K = 2$ **(b)** $K = 3$ **(c)** $K = 4$

Figure 2: An example of how a 30-node network could be partitioned into $K = \{2, 3, 4\}$ partitions with our proposed algorithm. The optimal number of partitions K depends on the generated network topology, the number, distributions and directions of the generated traffic flows.

the timeslot once at any given time. Specific details of these two approaches can be found in [9].

One obvious limitation of both S/PR and S/FR scheduling approaches is that the computational time required increases exponentially as the number of nodes increases. The joint routing and scheduling problem is defined as an NP-complete and bin-packing problem with additional constraints. The complexity of solving an ILP problem is influenced by the number of variables and constraints present. The S/PR and S/FR ILP formulations presented above include fundamental parameters that cannot be eliminated, such as the number of flows, the number of timeslots and the number of network links. In the next section, we present our proposed algorithm that seeks to address this limitation.

3 PROPOSED ALGORITHM FOR TSN SCHEDULING

In this section, we present the algorithm based on multi-level graph partitioning and ILP to address the challenge of exponentially-increasing computational time with the growth in network size. Our algorithm sub-divides a network graph into multiple sub-graphs using multi-level partitioning, then use an ILP solver to incrementally operate on each sub-graph.

Firstly, we interpret the network as an un-directed graph $G = (V, E)$, where V represent network nodes and E is a set of tuples which represent network links in the graph. We assume the property of all links, the length of the data flow and the period of the transmission over all links are identical. To model the network, Barabási–Albert topology [16] has been generated to simulate the network topology in our evaluation. The data flows generated in the network topology are cyclical flows that traverse from one node to another node within the same graph.

Our multi-level graph partitioning approach is a two-step process: *edge contraction* and *local search*. The edge contraction process reduces the size of the input network needed to reflect the global structure of the input network. Local search algorithms aim to improve the balance of the number of nodes between partitions. Using this method, a coarser graph can be obtained by collapsing adjacent vertices. The edge between two vertices is then collapsed. This step is defined as matching.

A matching graph is a set of edges, where no two of them are incident on the same vertex. The process of *maximal matching* results in a graph that contains all possible edges, where no two of them which are incident on the same vertex, hence reducing the size of the graph. A maximal matched graph contains the least number of edges.

After using multi-level graph partitioning to reduce the size of network graph, we use Gurobi [1] as our ILP solver to calculate the optimal scheduling time. We then use NetworkX [2] to model a network topology with N nodes (where N=30 to 100 in our evaluation), with time-triggered data flows being generated from source and destination nodes. Fig. 2 illustrates an example of how a 30-node network could be partitioned into $K = \{2, 3, 4\}$ partitions with our proposed algorithm. The optimal number of partitions K depends on the generated network topology, the flow number, flow distributions and the directions of the generated traffic flows.

Algorithm 1 describes our proposed algorithm. The weight on each link in the graph is first calculated based on the number of flows traverse through the link (*line 2*). Then the weights on every link are mapped back to the graph (*line 3*) the optimal k partitioning of the mapped graph is calculated (*line 4*). We use METIS [3] to perform the direct k-way partitioning (*line 5*).

After the partitioning process, the smaller graphs and their corresponding data flows are being served as inputs to the ILP solver. The ILP solver will iteratively solve the formulations and their constraints to generate the optimal time schedule for flows within each subset (*line 6-14*). The result is then the input of the cross-subnet flows (*line 15-17*). We use the computational time (time required to compute the optimal schedule) and the percentage of successfully scheduled flows as the key metrics for our performance evaluation.

[1] https://www.gurobi.com/
[2] https://networkx.org/
[3] https://metis.readthedocs.io/

Algorithm 1: Our proposed algorithm for TSN scheduling

```
Data: G, Flows
Result: SchedulingTable
Result: SchedulingTime
Initialisation (SchedulingTable)
W ← WeightsOnLinks (G)
MapWeightsToGraph (G,W)
K ← OptimalGraphPartition (G)
(Subsets, parts) ← METIS (G,K)
for i ← 0 to K do
    SubFlows ← MappingSubsetWithFlows (Subsets,Flows)
    Time ← Time()
end
Scheduling Time ← Max (Time)
FlowsNotInSubset ← Flows - SubFlows
for i ← 0 to parts do
    SchedulingTable ← ILPSolver (subset[n],
      SchedulingTable)
end
if FlowsNotInSubset == ∅ then
    Return SchedulingTable
else
    SchedulingTable ← ILPSolver (FlowsNotInSubset,
      SchedulingTable)
    Return SchedulingTable
end
```

4 EVALUATION RESULTS AND ANALYSIS

In this section, we present our evaluation results and analysis. We compared the performance of S/PR and S/FR scheduling approaches with and without applying our multi-level graph partitioning algorithm. The results generated when S/PR and S/FR are used without applying our algorithm served as our performance measurement benchmark. The platform running our simulations is a PC with an Intel Core i5 9400F/2.9 GHz processor with 16 GB RAM.

We evaluate the impact of network size (number of nodes), link capacity (number of timeslots in a single period) and the number of data flows on the computational time of the scheduling algorithms and the success rate of scheduled data flows. We evaluate our algorithm in three distinct scenarios, where the network has: (i) the number of nodes varying from 30 to 100; (ii) the number of timeslots varying from 5 to 30; and (iii) the number of data flows varying from 50 to 350.

4.1 The impact of network size

Fig. 3a presents the computational time required for S/PR and S/FR scheduling when different topology sizes are deployed, ranging from 30 to 100 nodes. The number of data flows in this scenario is fixed to 300 flows in a BA network topology with 10 timeslots per period. When our multi-level partitioning algorithm is not used, the overall computational and execution time of both the S/PR and S/FR scheduling processes are higher (regardless of the network size). Our algorithm significantly reduces the computational time as the number of nodes increases when compared with the standard S/PR and S/FR approaches. However, the computational time when using our approach only varies slightly as the number of nodes increases, demonstrating the efficacy of approach to reduce the computational time to almost the minimum time required.

In addition to reducing the computational time, our algorithm also improves the percentage of successfully-scheduled data flows, as described in Fig. 3d. It is noted that our algorithm shows a consistent 20% improvements when compared with standard S/PR and S/FR scheduling (without applying our algorithm).

4.2 The impact of the number of timeslots

Fig. 3b describes the computational time required for S/PR and S/FR scheduling when the number of timeslots within a period varies from 5 to 30. In this scenario, 300 data flows are deployed within a 50-node network topology. Without using our algorithm, the computational time for both S/PR and S/FR scheduling schemes is the least when the number of timeslots is 20 per period.

When our algorithm is applied, the computational time required reduces. It is observed that the computational time increases as the number of timeslots per period increases, closing the gap between using standard S/PR and S/FR, and our algorithm. This is due to the increasingly larger number of inputs into our ILP solver. In scenarios where the number of available timeslots is large (e.g., 20-30 timeslots per period), where more than 99% of data flows are successfully scheduled, as shown in Fig. 3e. Our algorithm demonstrated significant benefits when the available timeslots are limited. For example, when we have five timeslots per period, our algorithm managed to enhance S/PR and S/FR scheduling schemes to achieve 15% higher success rate.

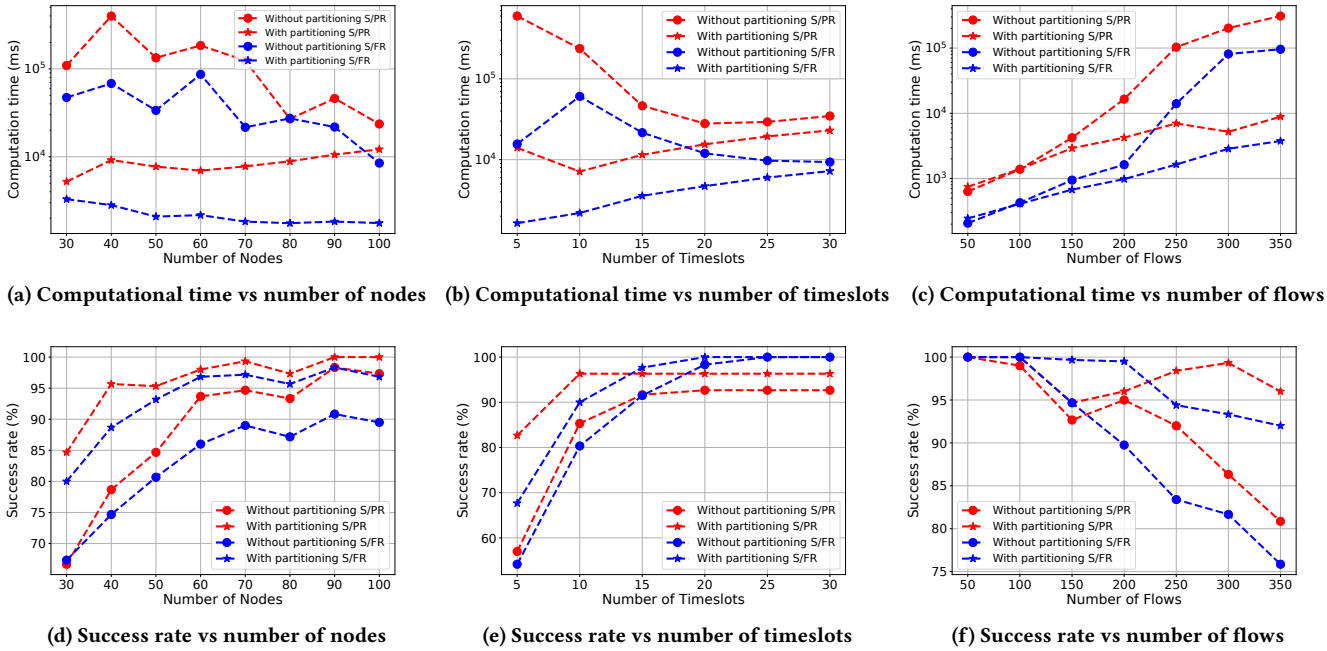
4.3 The impact of the number of data flows

Fig. 3c shows the computational time required for S/PR and S/FR scheduling when the number of time-triggered data flows varies from 50 to 350. The network size is fixed to 50 nodes. Without using our algorithm, the computational time of both S/PR and S/FR scheduling increases almost exponentially as the number of data flows increases. When our algorithm is applied, the computational time increases linearly, and is significantly lower when compared with standard S/PR and S/FR scheduling. In terms of success rate of the scheduled data flows, our algorithm shows significant improvements where it achieves a 20% higher success rate when the number of flows is large (e.g., 350 flows).

It is important to note that the results presented in Fig. 3 have included the optimal number of partitions (optimal K) in the specified scenarios. The optimal number of partitions vary depending on the generated network topology, the number, distributions and directions of the generated traffic flows. The results of optimal K across these scenarios are not presented due to space constraints.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have designed and proposed a novel multi-level graph partitioning approach for optimal scheduling in TSN-enabled DTNs, where the number of nodes, the number of available timeslots per period and the number of data flows are typically large. This is particularly evident in DTNs where the mappings between



(a) Computational time vs number of nodes **(b) Computational time vs number of timeslots** **(c) Computational time vs number of flows**

(d) Success rate vs number of nodes **(e) Success rate vs number of timeslots** **(f) Success rate vs number of flows**

Figure 3: Computational time required and the success rate of the scheduled flows vs the number of nodes, timeslots and flows.

physical objects and virtual twins form a many-to-many communication relationship. We use extensive simulations to demonstrate the effectiveness of our algorithm. Our results have demonstrated that our algorithm can (i) reduce the computational time required to calculate the optimal transmission schedule for TSN data flows, and (ii) improve the rate of successfully scheduled data flows. When applied to both S/PR and S/FR scheduling techniques, our algorithm is able to achieve up to five times reduction in computation time and up to 20% improvements in data flows success rate when a large number of data flows is present.

Our current work is based on simplified assumptions of DTNs, including the network configurations, flow distributions, and communication requirements between physical objects and virtual twins. As more DTN data becomes publicly available, our future work will incorporate more realistic simulations and experiments to further enhance our proposed algorithm. Besides, our current work only focuses on S/PR and S/FR scheduling techniques, and evaluated the computational time as a whole. Our future work will include integrating our algorithm with other scheduling schemes for TSN and DTNs, and identifying their performance bottleneck; characterising the latency requirements of various DT applications; improving the optimality of graph partitions, to meet the diverse QoS requirements of heterogeneous (a mix of non-DT and DT) communication nodes in modern industrial networks that are increasingly converging.

REFERENCES

[1] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. Industry 4.0. *Business & information systems engineering*, 6(4):239–242, 2014.

[2] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund. Industrial internet of things: Challenges, opportunities, and directions. *IEEE Transactions on Industrial Informatics*, 14(11):4724–4734, Jul 2018.

[3] Michael Grieves and John Vickers. Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems. In *Transdisciplinary perspectives on complex systems*, pages 85–113. Springer, 2017.

[4] Yiwen Wu, Ke Zhang, and Yan Zhang. Digital twin networks: A survey. *IEEE Internet of Things Journal*, 8(18):13789–13804, 2021.

[5] Norman Finn. Introduction to time-sensitive networking. *IEEE Communications Standards Magazine*, 2(2):22–28, 2018.

[6] Jörg Christian Kirchhof, Judith Michael, Bernhard Rumpe, Simon Varga, and Andreas Wortmann. Model-driven digital twin construction: synthesizing the integration of cyber-physical systems with their information systems. In *Proceedings of the 23rd ACM/IEEE International Conference on Model Driven Engineering Languages and Systems*, pages 90–101, 2020.

[7] Lucia Lo Bello and Wilfried Steiner. A perspective on ieee time-sensitive networking for industrial communication and automation systems. *Proceedings of the IEEE*, 107(6):1094–1120, 2019.

[8] W. Steiner. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *2010 31st IEEE Real-Time Systems Symposium*, pages 375–384, Nov 2010.

[9] Naresh Ganesh Nayak, Frank Dürr, and Kurt Rothermel. Time-sensitive software-defined network (TSSDN) for real-time applications. In *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, pages 193–202, New York, NY, USA, Oct 2016.

[10] Voica Gavriluț and Paul Pop. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In *2018 14th IEEE International Workshop on Factory Communication Systems*, pages 1–4, Jun 2018.

[11] Peter Sanders and Christian Schulz. Engineering multilevel graph partitioning algorithms. In *European Symposium on Algorithms*, pages 469–480. Springer, 2011.

[12] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.

[13] Paolo Bellavista, Carlo Giannelli, Marco Mamei, Matteo Mendula, and Marco Picone. Application-driven network-aware digital twin management in industrial edge environments. *IEEE Trans. Ind. Informat*, 17(11):7791–7801, 2021.

[14] Yunlong Lu, Xiaohong Huang, Ke Zhang, Sabita Maharjan, and Yan Zhang. Communication-efficient federated learning for digital twin edge networks in industrial iot. *IEEE Transactions on Industrial Informatics*, 17(8):5709–5718, 2020.

[15] Alexandra Henzinger, Alexander Noe, and Christian Schulz. ILP-based local search for graph partitioning. *ACM J. Exp. Algorithmics*, 25, Jul 2020.

[16] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.