MRDAG-Gen: Multi-rate DAG Generator

***Abstract*—TODO:**

***Index Terms*—TODO:**

I. Introduction

Real-time systems such as self-driving systems must be successfully executed while meeting various requirements such as output within a pre-determined time (i.e., deadline), low power consumption, and resource constraints [1]–[3]. To meet these constraints, there has been much research on task allocation and scheduling [4]–[6], as well as on analyzing the end-to-end latency and the response time of a system [7]–[9]. Systems are becoming larger and more complex every year, and such studies use models that represent the complex dependencies and parallelism of tasks in the system, such as a directed acyclic graph (DAG).

DAGs are used in many allocation, scheduling, and latency analysis studies [10]–[12] because they express the flow of processing from system input to output and can represent various kinds of information such as dependencies between tasks, task execution time, and execution period. To evaluate the performance of proposed methods in these studies, it is important to compare them with existing methods using task sets. Then, in the evaluation of methods using DAGs, randomly generated DAGs are used to ensure objectivity and to demonstrate generality [13]–[15].

To ease such evaluations, random DAG generation tools such as the task graph for free (TGFF) [16] and GGen [17] have been proposed and utilized in the latest publications [18]–[21]. These tools allow the user to parametrically specify the shape of the DAG (e.g., number of nodes, in-degree and out-degree per node) and the properties assigned to nodes and edges (e.g., execution time, execution period, communication time). Furthermore, because these tools use a pseudo-random number generator, other researchers can easily reproduce the DAG set by specifying the same options. However, TGFF and GGen have been proposed in 1999 and 2010, respectively, and cannot meet the requirements for multi-rate DAGs considering the state-of-the-art real-time systems.

Since embedded systems in automobiles and avionics, as well as self-driving systems, contain multiple tasks that operate at different periods (e.g., sensors [22], localization [14] and angle synchronous [23]), research targeting multi-rate DAGs is becoming increasingly important [8], [15]. In studies of such multi-rate DAGs, not only the shape of the DAG but also the ratio of execution time to task execution period has a significant impact on the performance of the method (e.g., implicit deadline [24], [25] and task utilization [26], [27]). However, TGFF cannot generate multi-rate DAGs, and GGen can only randomly set the period and the execution time to nodes. Therefore, most authors who consider multi-rate DAGs have their implementation of a random DAG set [26]–[29]. It is laborious for researchers to prepare their own set of random DAGs, and further reduces the reliability and reproducibility of the evaluation results.

To solve these problems, this paper proposes a random DAG generation tool called a multi-rate DAG generator (MRDAG-Gen) that meets the requirements of state-of-the-art research. MRDAG-Gen extends existing DAG generation methods and provides a flexible evaluation platform. Since MRDAG-Gen uses a pseudo-random number generator, other researchers can reproduce the DAG sets used in the evaluation by specifying the same options. In addition, MRDAG-Gen supports researchers by providing batch generation of all random DAG sets at different parameters and the functionality to visualize scheduling results.

Contributions: Our primary contributions are summarized as follows.

- MRDAG-Gen provides a flexible DAG set by adding parameters to existing DAG generation methods and new chain-based generation methods.
- MRDAG-Gen automatically sets properties that meet implicit deadlines and utilization constraints.
- MRDAG-Gen reduces implementation effort through a batch generation of random DAG sets and visualization of scheduling results.

The remainder of the paper is organized as follows. Section II describes a system model. Section III explains the design and implementation of MRDAG-Gen. Section IV demonstrates the use of MRDAG-Gen based on case studies. Section V compares the performance of the proposed method with existing methods. Section VI compares MRDAG-Gen with existing random DAG generation methods. Finally, Section VII presents the conclusions and future work.

II. System model

TODO:

III. Design and Implementation

TODO:

IV. Case study

TODO:

V. Evaluation

TODO:

VI. Related work

TODO:

VII. Conclusion

TODO:

References

[1] Ryotaro Koike and Takuya Azumi. Federated scheduling in clustered many-core processors. In *Proc. of DS-RT*, 2021.

[2] Debabrata Senapati, Arnab Sarkar, and Chandan Karfa. HMDS: A makespan minimizing DAG scheduler for heterogeneous distributed systems. *TECS*, 2021.

[3] Avinash Kaur, Parminder Singh, Ranbir Singh Batth, and Chee Peng Lim. Deep-Q-learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud. *J. Software. Pract. Exper.*, 2020.

[4] Shingo Igarashi, Takuro Fukunaga, and Takuya Azumi. Accurate contention-aware scheduling method on clustered many-core platform. *J. IPSJ*, 2021.

[5] Ali Asghari, Mohammad Karim Sohrabi, and Farzin Yaghmaee. Online scheduling of dependent tasks of cloud's workflows to enhance resource utilization and reduce the makespan using multiple reinforcement learning-based agents. *J. Soft. Comput.*, 2020.

[6] Zhao Tong, Xiaomei Deng, Hongjian Chen, Jing Mei, and Hong Liu. QL-HEFT: a novel machine learning scheduling scheme base on cloud computing environment. *J. Neural. Comput. Appl.*, 2020.

[7] Yuqing Yang and Takuya Azumi. Exploring real-time executor on ROS 2. In *Proc. of ICESS*, 2020.

[8] Alix Kordon and Ning Tang. Evaluation of the age latency of a real-time communicating system using the LET paradigm. In *Proc. of ECRTS*, 2020.

[9] Peng Chen, Hui Chen, Jun Zhou, Di Liu, Shiqing Li, Weichen Liu, Wanli Chang, and Nan Guan. Partial order based non-preemptive communication scheduling towards real-time networks-on-chip. In *Proc. of SAC*, 2021.

[10] Hyunjong Choi, Yecheng Xiang, and Hyoseung Kim. PiCAS: New design of priority-driven chain-aware scheduling for ROS2. In *Proc. of RTAS*, 2021.

[11] Viet Anh Nguyen, Damien Hardy, and Isabelle Puaut. Cache-conscious off-line real-time scheduling for multi-core platforms: algorithms and implementation. *J. Real-Time Syst.*, 2019.

[12] Tobias Klaus, Matthias Becker, Wolfgang Schröder-Preikschat, and Peter Ulbrich. Constrained data-age with job-level dependencies: How to reconcile tight bounds and overheads. In *Proc. of RTAS*, 2021.

[13] Gaoyang Dai, Morteza Mohaqeqi, and Wang Yi. Timing-anomaly free dynamic scheduling of periodic dag tasks with non-preemptive nodes. In *Proc. of RTCSA*, 2021.

[14] Micaela Verucchi, Mirco Theile, Marco Caccamo, and Marko Bertogna. Latency-aware generation of single-rate DAGs from multi-rate task sets. In *Proc. of RTAS*, 2020.

[15] Mario Günzel, Niklas Ueter, and Jian-Jia Chen. Suspension-aware fixed-priority schedulability test with arbitrary deadlines and arrival curves. In *Proc. of RTSS*, 2021.

[16] R. P. Dick, D. L. Rhodes, and W. Wolf. TGFF: task graphs for free. In *Proc. of Workshop on CODES/CASHE*, 1998.

[17] Daniel Cordeiro, Grégory Mounié, Swann Perarnau, Denis Trystram, Jean-Marc Vincent, and Frédéric Wagner. Random graph generation for scheduling simulations. In *Proc. of SIMUTools*, 2010.

[18] Penghao Sun, Zehua Guo, Junchao Wang, Junfei Li, Julong Lan, and Yuxiang Hu. Deepweave: Accelerating job completion time with deep reinforcement learning-based coflow scheduling. In *Proc. of IJCAI*, 2021.

[19] Chun-Hsian Huang. HDA: Hierarchical and dependency-aware task mapping for network-on-chip based embedded systems. *J. Syst. Archit.*, 2020.

[20] Benjamin Rouxel, Stefanos Skalistis, Steven Derrien, and Isabelle Puaut. Hiding communication delays in contention-free execution for SPM-based multi-core architectures. In *Proc. of ECRTS*, 2019.

[21] Kun Cao, Junlong Zhou, Peijin Cong, Liying Li, Tongquan Wei, Mingsong Chen, Shiyan Hu, and Xiaobo Sharon Hu. Affinity-driven modeling and scheduling for makespan optimization in heterogeneous multiprocessor systems. *TCAD*, 2018.

[22] Liu Shaoshan, Yu Bo, Guan Nan, Dong Zheng, and Akesson Benny. Industry challenge. In *Proc. of Industry Session (part of RTSS)*, 2021.

[23] Arne Hamann, Dakshina Dasari, Simon Kramer, Michael Pressler, and Falk Wurst. Communication centric design in complex automotive embedded systems. In *Proc. of ECRTS*, 2017.

[24] Niklas Ueter, Mario Günzel, Georg von der Brüggen, and Jian-Jia Chen. Hard real-time stationary gang-scheduling. In *Proc. of ECRTS*, 2021.

[25] Youngeun Cho, Dongmin Shin, Jaeseung Park, and Chang-Gun Lee. Conditionally optimal parallelization of real-time DAG tasks for global EDF. In *Proc. of RTSS*, 2021.

[26] Suhail Nogd, Geoffrey Nelissen, Mitra Nasri, and Björn B Brandenburg. Response-time analysis for non-preemptive global scheduling with fifo spin locks. In *Proc. of RTSS*, 2020.

[27] Kecheng Yang and Zheng Dong. Mixed-criticality scheduling in compositional real-time systems with multiple budget estimates. In *Proc. of RTSS*, 2020.

[28] Sergey Voronov, Stephen Tang, Tanya Amert, and James H Anderson. AI meets real-time: Addressing real-world complexities in graph response-time analysis. In *Proc. of RTSS*, 2021.

[29] Zheng Dong and Cong Liu. An efficient utilization-based test for scheduling hard real-time sporadic dag task systems on multiprocessors. In *Proc. of RTSS*, 2019.