My research

Abhishek Singh

This document contains five examples of my research. Each instance focuses on ensuring properties such as timeliness and security in computer systems through modeling and analysis. In many cases, the proposed models and algorithms are surprising and challenge established ideas. In the first example, I have used the theory of cutting planes to explain the behavior of a class of well-established verification algorithms, and I have shown how to construct faster algorithms using this insight. Similarly, in the second and third examples, I have used an unorthodox parameterized definition of computational efficiency to study the analyzability of timed models of computer systems. In the third and fourth examples, I have investigated the balance between the expressiveness of models of real-time systems so that they can express security or dataflow constraints and their algorithmic analyzability. In the last example, I have proposed techniques to improve the performance of systems that contain general-purpose components and components with timeliness constraints. Most results are a mix of modeling of computer systems (using theories and first principles), analysis of models (using ideas from parameterized algorithms, combinatorial optimization, integer programming, mixed-integer nonlinear programming), and empirical performance evaluations (using workload generation).

Background

> [...] reactive systems are everywhere. From microwave ovens and digital watches, through man/machine based software systems, silicon chips, robots and communication networks, all the way to computer operating systems, complex industrial plants, avionics systems and the like.
>
> D. Harel and A. Pnueli

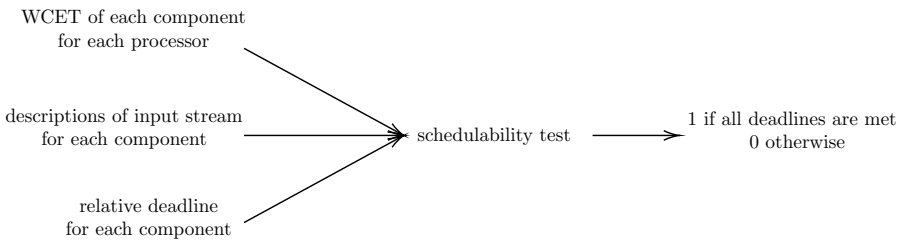
A component in a reactive system consumes a stream of input events and produces a stream of output events:

A safety-critical component in a reactive system is often required to satisfy a real-time property specified in terms of a *relative deadline* $D > 0$: for any input event i_n, the corresponding output event o_n must occur within D units

of physical time. Any collection of such components is a real-time system. The theory of real-time systems is concerned with the following problem:

> Given a description of a finite collection of components in a programming language and a description of the physical hardware architecture (processors, memory, communication, etc.), how do we implement the software components on the hardware so that the real-time requirements of all components are satisfied?

The computation of the worst-case execution time (WCET) of a component on a processor is a necessary step in solving the problem. Techniques such as combining static analysis of the component with some model of the processor are often used to compute the WCET. See reference [1] for an overview. The next step is to characterize the timing behavior of the input stream. For instance, one may specify that any two input events are separated by a minimum duration in time (such an input stream is said to be *sporadic* [2]). The final step is to come up with a *scheduling policy* and to solve the associated *schedulability problem*. A scheduling policy is an online algorithm that determines the allocation of resources, e.g., processors and shared variables, to the software components during runtime. In simple scenarios, preemptive priority-based scheduling policies such as FP (fixed priority) and EDF (earliest deadline first) are used. In the schedulability problem, we are given the WCETs and relative deadlines of the components and a concise description of the timing behavior of the input streams, and we must decide whether the real-time requirements of the components will be satisfied in all timing behaviors of the system, where each timing behavior is a function of the input streams, the components, and the scheduling policy. An algorithm that solves the schedulability problem is called a *schedulability test*.



Classical timed models of computer systems and schedulability tests may be found in reference [3].

1 Efficient schedulability tests using cutting-plane theory

Fixed-point iteration algorithms such as RTA (response time analysis) and QPA (quick processor-demand analysis) are arguably the most popular algorithms for solving elementary schedulability problems [4, 5, 6]. These algorithms are quite fast in practice, especially when they use a good initial guess for the fixed point [7, 8, 9]. I have shown that these algorithms are part of a larger class of cutting-plane methods, and I have discovered an algorithm with the optimal

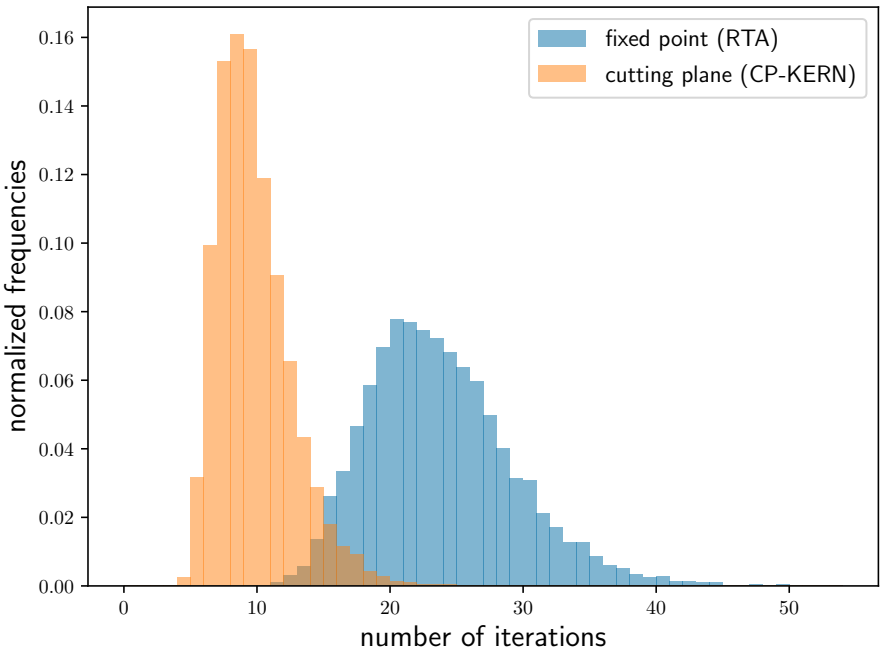


Figure 1: Empirical evidence for the superior convergence rate of the cutting-plane algorithm, confirming the theoretical results. See reference [10] for details.

convergence rate in this class [10]. My work has led to the logical unification of two seemingly disparate approaches toward solving the problems, i.e., fixed-point methods and cutting-plane methods. Moreover, the new algorithm is the state of the art in terms of convergence rate and is faster than fixed-point iteration algorithms on challenging instances of the schedulability problem.

Now that schedulability tests are connected to cutting-plane theory, we can pursue a deeper understanding of schedulability problems by using advanced tools from the theory. For instance, instead of traditional textbook cutting planes, we can generate deeper cuts that define facets of the schedulability polytope; this is known as the *polyhedral approach* [11].

Since my work shows that some general-purpose algorithms can be more effective than domain-specific algorithms such as RTA and QPA, it raises the question: are there other general-purpose algorithms, e.g., for arithmetic or logic, that can efficiently solve the schedulability problem?

The questions mentioned in the last two paragraphs exemplify research directions that anyone can investigate immediately in this space. This line of research will yield more efficient algorithms that will empower the model-based development of reliable reactive systems.

2 Understanding analyzability using parameterized complexity

If we equate computational efficiency with "worst-case polynomial time wrt input size", then the schedulability problem is intractable even in elementary cases, e.g., preemptive FP sporadic uniprocessor systems. See reference [12] for an overview of complexity results on preemptive uniprocessor schedulability and reference [13] for details about efficiency and polynomial time. In defiance of the theoretical intractability, researchers have discovered schedulability tests that work well in practice, e.g., the fixed-point iteration and cutting-plane algorithms mentioned in the previous section.

Researchers have observed similar gaps between theoretical intractability and practical efficiency for algorithms such as simplex for linear programming and conflict-driven clause learning for propositional satisfiability. See reference [14] for more examples of this phenomenon. Paradigms such as parameterized algorithms [15, 16] and algorithm engineering [17, 18] are developing rapidly to address this gap between theory and practice.

Parameters are first-class citizens in the theory of parameterized algorithms: we cannot discuss the efficiency of an algorithm without reference to a parameter of the problem. For instance, we can study algorithms for the integer programming problem with respect to the parameters such as "no. of variables" and "no. of constraints". We equate efficiency with "worst-case polynomial time wrt input size" modulo the effects of the parameter on efficiency. We ignore the effects of the parameter because we assume that it is small. See references [15, 16] for more details.

We have studied fundamental schedulability problems using the framework provided by parameterized algorithms [19]. Our work has led to a better understanding of the tractability of elementary schedulability problems with respect to various natural choices of parameters. I have also posed open parameterized complexity classification problems to the real-time systems community [20]. By using techniques and results in the theory of parameterized algorithms, I have improved existing schedulability tests and also proposed new schedulability tests.

Although I have applied the framework of parameterized algorithms to schedulability problems in real-time systems theory, reducing the gap between theoretical intractability and practical efficiency is quite a general idea, and it can be studied in any area at the intersection of systems and algorithmic theory.

3 Addressing security and timeliness concerns through setup times

When components run on the same processor, information may leak between them through the state of the processor's memory cache. Such side channels can compromise the security of critical components in the system. An expensive yet effective countermeasure is to flush the cache on every context switch from one process to another [21]. If the components also have real-time requirements,

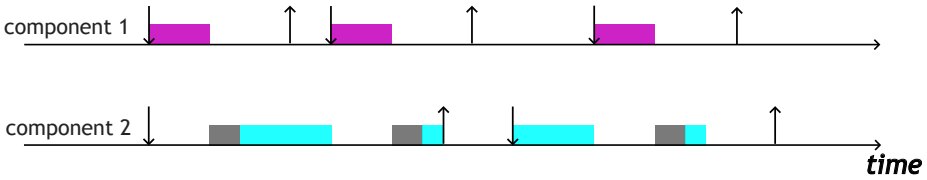


Figure 2: Down (resp., up) arrows represent input events (resp., deadlines for output events). The bright colored bars represent components executing on the single processor, and the grey bar is the cache flush time (setup time in the model) because we wish to remove side channels from component 1 to component 2.

then frequent cache flushes can make it harder to meet the requirements: more intelligent scheduling policies than EDF and FP are needed here.

I have modeled the problem of meeting deadlines when the cache-flushing policy is active as a job scheduling problem with setup times [22]. Scheduling problems with setup times are usually studied in the operations research (OR) community, and several classical algorithmic and complexity results are available. See, for instance, reference [23] for an overview. Now, we can use these classical algorithmic and complexity results to understand these timeliness problems in computer systems.

In my model of the cache-flushing policy, I have observed that the number of families and the largest setup time are small in most natural systems. Therefore, it makes sense to study the setup times problem using the framework of parameterized algorithms, where the number of families and the largest setup time are natural candidates for parameters. I have discovered new parameterized intractability and tractability results for these parameters. While these discoveries are important, many more developments are needed to understand the parameterized landscape for the setup times problem and its variants. The computationally tractable variants have the potential to serve as building blocks for better periodic real-time systems.

In summary, I have examined the expressive OR problem of setup times using the framework of parameterized algorithms, where I chose the parameters by observing natural computer systems. In general, expressive OR scheduling problems can help model complex interactions between scheduling decisions, low-level architectural quirks and high-level objectives such as energy management and security. Furthermore, complex systems problems can inform how to approach OR scheduling problems (or other fundamental combinatorial problems) from nuanced parameterized perspectives.

4 Dataflow models

In reactive systems where precise control over data flow is essential, designers often model individual components as synchronous dataflow graphs (SDFGs). See reference [24, Section 6.3] for an overview of dataflow models. If each SDFG component also has real-time requirements, then it is natural to measure the WCET of each node in the SDFG separately. We need new scheduling policies

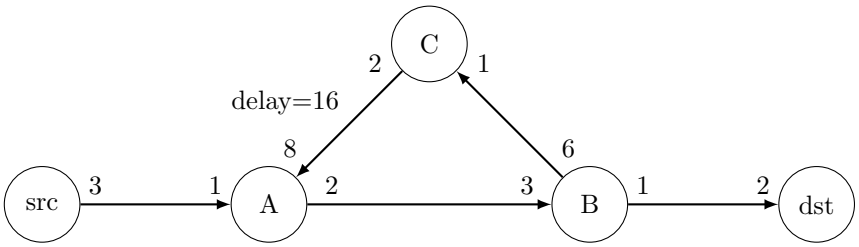


Figure 3: An SDFG component with nodes and edges labeled with production, consumption, and delay values. A real-time requirement may be specified between the source node and the target node.

and schedulability tests in this setting.

We have shown that the problem of scheduling SDFG components on a uniprocessor can be reduced to the problem of scheduling nodes in the components by assigning a deadline to each node [25]. The reduction allows us to ignore the graph structure inside each component: we use EDF to schedule the nodes and the schedulability test for EDF to check the schedulability of the system. In later work, we have improved the reduction algorithm and shown that it runs in polynomial time [26].

We have shown that when scheduling a restricted class of SDFG components (homogeneous SDFGs) on m processors, global-EDF can be chosen as the scheduling policy and have proposed a sufficient schedulability test that satisfies the tight speedup bound of $(2 - 1/m)$ [27]. See reference [28] for understanding speedup bounds and techniques for schedulability analysis of graph-based workloads on multiprocessors.

5 Mixing real-time and general-purpose components

For many reactive systems, the safety-critical components have real-time requirements, while other general-purpose components are only required to be responsive. For such hybrid systems, designers often use an aperiodic server to serve the general-purpose components at a high priority while simulating the timing behavior of an artificial real-time component at that priority [29]. This results in smaller response times for the general-purpose components. Moreover, we can use traditional schedulability tests to analyze the system since it effectively contains only real-time components from a timing perspective. We must pick the parameters of the artificial real-time components to increase the responsiveness of the general-purpose components while satisfying real-time requirements.

It is known from empirical studies that the general-purpose components are more responsive if the server has a high priority, a large budget, and a large utilization [30, Sec. 4.4]. We have proposed a new class of aperiodic servers that can achieve larger budgets and utilizations by simulating multiple real-time components for fixed-priority (FP) systems [31]. We have proposed a linear-

time algorithm for choosing the parameters of these new aperiodic servers for a restricted class of FP systems and an MINLP (mixed-integer nonlinear program) for the general case.

There are many open research ideas in this space, such as formal verification of server designs and improvement of server designs using ideas from control theory and machine learning.

References

[1] Reinhard Wilhelm, Jakob Engblom, Andreas Ermedahl, Niklas Holsti, Stephan Thesing, David Whalley, Guillem Bernat, Christian Ferdinand, Reinhold Heckmann, Tulika Mitra, Frank Mueller, Isabelle Puaut, Peter Puschner, Jan Staschulat, and Per Stenström. The worst-case execution-time problem—overview of methods and survey of tools. *ACM Transactions on Embedded Computing Systems*, 7(3):36:1–36:53, May 2008.

[2] Aloysius Ka-Lau Mok. *Fundamental design problems of distributed systems for the hard-real-time environment.* Thesis, Massachusetts Institute of Technology, 1983. Accepted: 2005-08-04T22:02:06Z.

[3] C. L. Liu and James W. Layland. Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM*, 20(1):46–61, January 1973.

[4] M. Joseph and P. Pandya. Finding Response Times in a Real-Time System. *The Computer Journal*, 29(5):390–395, January 1986.

[5] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[6] Fengxiang Zhang and Alan Burns. Schedulability Analysis for Real-Time Systems with EDF Scheduling. *IEEE Transactions on Computers*, 58(9):1250–1258, September 2009.

[7] M. Sjodin and H. Hansson. Improved response-time analysis calculations. In *Proceedings 19th IEEE Real-Time Systems Symposium (Cat. No.98CB36279)*, pages 399–408, December 1998.

[8] R.J. Bril, W.F.J. Verhaegh, and E.-J.D. Pol. Initial values for online response time calculations. In *15th Euromicro Conference on Real-Time Systems, 2003. Proceedings.*, pages 13–22, July 2003.

[9] Robert I. Davis, Attila Zabos, and Alan Burns. Efficient Exact Schedulability Tests for Fixed Priority Real-Time Systems. *IEEE Transactions on Computers*, 57(9):1261–1276, September 2008.

[10] Abhishek Singh. Cutting-plane algorithms for preemptive uniprocessor scheduling problems. *Real-Time Systems*, August 2023.

[11] Manfred Padberg and Giovanni Rinaldi. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems.

SIAM Review, 33(1):60–100, 1991. Publisher: Society for Industrial and Applied Mathematics.

[12] Pontus Ekberg and Wang Yi. Complexity of Uniprocessor Scheduling Analysis. In Yu-Chu Tian and David Charles Levy, editors, *Handbook of Real-Time Computing*, pages 1–18. Springer, Singapore, 2020.

[13] Walter Dean. Computational Complexity Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2021 edition, 2021.

[14] Tim Roughgarden. Introduction. In Tim Roughgarden, editor, *Beyond the Worst-Case Analysis of Algorithms*, pages 1–24. Cambridge University Press, 1 edition, December 2020.

[15] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford, 2006.

[16] Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Daniel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer Publishing Company, Incorporated, 1st edition, 2015.

[17] Peter Sanders. Algorithm Engineering – An Attempt at a Definition. In Susanne Albers, Helmut Alt, and Stefan Näher, editors, *Efficient Algorithms: Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday*, Lecture Notes in Computer Science, pages 321–340. Springer, Berlin, Heidelberg, 2009.

[18] Matthias Müller-Hannemann and Stefan Schirra, editors. *Algorithm Engineering: Bridging the Gap between Algorithm Theory and Practice*, volume 5971 of *Lecture Notes in Computer Science*. Springer, Berlin, Heidelberg, 2010.

[19] Sanjoy Baruah, Pontus Ekberg, and Abhishek Singh. Fixed-Parameter Analysis of Preemptive Uniprocessor Scheduling Problems. In *2022 IEEE Real-Time Systems Symposium*, page 12, 2022.

[20] Abhishek Singh. Models and Algorithms for Real-Time Systems. *McKelvey School of Engineering Theses & Dissertations*, May 2023.

[21] Dag Arne Osvik, Adi Shamir, and Eran Tromer. Cache Attacks and Countermeasures: The Case of AES. In David Pointcheval, editor, *Topics in Cryptology – CT-RSA 2006*, Lecture Notes in Computer Science, pages 1–20, Berlin, Heidelberg, 2006. Springer.

[22] Abhishek Singh. On the intractability of preemptive single-machine job scheduling with release times, deadlines, and family setup times. *Information Processing Letters*, 179:106305, January 2023.

[23] Chris N. Potts and Mikhail Y. Kovalyov. Scheduling with batching: A review. *European Journal of Operational Research*, 120(2):228–249, January 2000.

[24] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to Embedded Systems: A Cyber-Physical Systems Approach*. The MIT Press, 2nd edition, November 2016.

[25] Abhishek Singh, Pontus Ekberg, and Sanjoy Baruah. Applying Real-Time Scheduling Theory to the Synchronous Data Flow Model of Computation. In Marko Bertogna, editor, *29th Euromicro Conference on Real-Time Systems (ECRTS 2017)*, volume 76 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 8:1–8:22, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISSN: 1868-8969.

[26] Abhishek Singh, Pontus Ekberg, and Sanjoy Baruah. Uniprocessor scheduling of real-time synchronous dataflow tasks. *Real-Time Systems*, 55(1):1–31, January 2019.

[27] Abhishek Singh and Sanjoy Baruah. Global EDF-Based Scheduling of Multiple Independent Synchronous Dataflow Graphs. In *2017 IEEE Real-Time Systems Symposium (RTSS)*, pages 307–318, December 2017. ISSN: 2576-3172.

[28] Vincenzo Bonifaci, Alberto Marchetti-Spaccamela, Sebastian Stiller, and Andreas Wiese. Feasibility Analysis in the Sporadic DAG Task Model. In *2013 25th Euromicro Conference on Real-Time Systems*, pages 225–233, July 2013. ISSN: 2377-5998.

[29] John P. Lehoczky, Lui Sha, and Jay K. Strosnider. Enhanced Aperiodic Responsiveness in Hard Real-Time Environments. In *Proceedings of the 8th IEEE Real-Time Systems Symposium (RTSS '87), December 1-3, 1987, San Jose, Califorinia, USA*, pages 261–270. IEEE Computer Society, 1987.

[30] G. Bernat and A. Burns. New results on fixed priority aperiodic servers. In *Proceedings 20th IEEE Real-Time Systems Symposium (Cat. No.99CB37054)*, pages 68–78, December 1999. ISSN: 1052-8725.

[31] Abhishek Singh and Sanjoy Baruah. Dimensions of Fixed-Priority Aperiodic Servers. In *Proceedings of the 31st International Conference on Real-Time and Network Systems*, page 11, 2023.