technische universität dortmund

computer science 12

Master Thesis

Dirichlet-Rescale Algorithm

Mario Günzel
Prof. Dr. Jian-Jia Chen

Otto-Hahn Str. 16
Technische Universität Dortmund
Email: mario.guenzel@tu-dortmund.de
January 16, 2024

One fundamental ingredient in the examination of real-time systems is the application of schedulability tests. However, when there are several schedulability tests available, this immediately raises the question:

Which schedulability test performs better?

Especially if there is no dominance between schedulability tests, synthetic task sets are generated and utilized to empirically evaluate the performance.

In the literature, several methods of generating task sets are widely used, e.g., *UUnifast* proposed in 2005, *UUnifast-Discard* proposed in 2009 and *RandFixedSum* proposed in 2006 (cf. [1]). Recently, in 2020, a new method called *Dirichlet-Rescale* (DRS) algorithm has been proposed by Griffin et al. [2], which seems to be more efficient and powerful than the previous methods.

The main idea of the DRS algorithms is to rescale simplices that fall outside a desired region. Figure 1 illustrates the main idea of this approach. In the middle is the desired region, i.e., each point in this desired region is equivalent to one generated task set. Outside we can see triangles. If the generated vector (e.g. point P^0) lies for example in the bottom left simplex (the yellow one), then the simplex is rescaled to match the standard simplex. This moves the point P^0 to P^1. Repeating this procedure several times, we obtain a value P^3 inside the desired area.

The DRS algorithm is widely applicable to several scenarios besides the standard task model if necessary enhancement or integration of the algorithm can be provided. That includes for example mixed criticality systems, multicore systems, typical and worst-case execution times, and resource locking protocols. Furthermore, the DRS algorithm is especially interesting to self-suspending task models, since at the moment there is no "de facto" synthetic benchmarks to evaluate scheduling algorithms and schedulbility tests.

In this master thesis, we would like to unleash the real power of this DRS algorithm for deciding workload parameters for self-suspending task sets. We foresee that

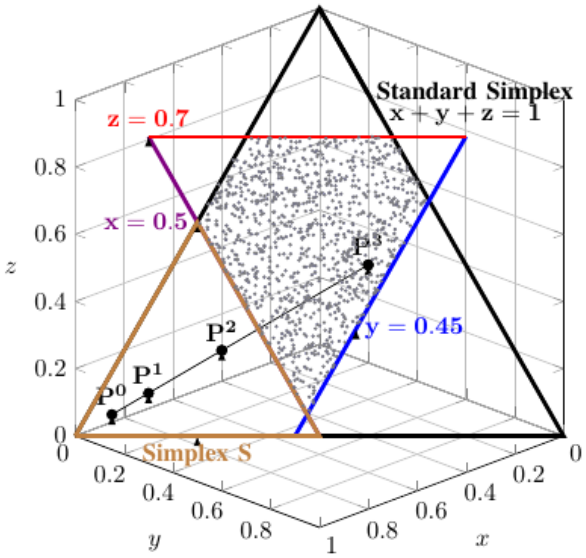

Figure 1: The output of Dirichlet-Rescale over 1000 runs in [2].

the DRS algorithm will be very effective and could provide a new perspective on the performance of some schedulability tests. Moreover, the experience of adapting the DRS algorithm can be generalized a series of topics in the investigation of real-time systems.

In this thesis, students will get familiar with the Dirichlet-Rescale (DRS) algorithm and explore its potential for the synthesization self-suspending task sets. The algorithm should be implemented and released as part of the evaluation framework [3]. Using this framework, a comparison with previous benchmarks will be conducted using already implemented schedulability tests.

Required Skills:

- Knowledge about real-time systems
- Basic knowledge about numerical simulations
- Comfortable in Python programming

Acquired Skills after the thesis:

- Knowledge about self-suspension
- Deep understanding of workload generators
- Experience of research campaigns

technische universität dortmund

computer science 12

Master Thesis

Dirichlet-Rescale Algorithm

Mario Günzel
Prof. Dr. Jian-Jia Chen

Otto-Hahn Str. 16
Technische Universität Dortmund
Email: mario.guenzel@tu-dortmund.de
January 16, 2024

References:

[1] Paul Emberson, Roger Stafford, and Robert I. Davis. "Techniques for the synthesis of multiprocessor tasksets." proceedings 1st International Workshop on Analysis Tools and Methodologies for Embedded and Real-time Systems (WATERS 2010). 2010. (Download preprint).

[2] David Griffin, Iain Bate, and Robert I. Davis. "Generating utilization vectors for the systematic evaluation of schedulability tests." 2020 IEEE Real-Time Systems Symposium (RTSS). IEEE, 2020. (Download preprint).

[3] LS12 DEAS Group. Evaluation Framework for Self-Suspending Task Systems. (Github repository).