On-line Scheduling of Target Sensitive Periodic Tasks with the Gravitational Task Model

Raphael Guerra, Gerhard Fohler
Technische Universität Kaiserslautern, Germany
{guerra,fohler}@eit.uni-kl.de

***Abstract*—Target sensitive tasks have an execution window for feasibility and must execute at a target point in time for maximum utility. In the gravitational task model, a task can express a target point, and the utility decay as a function of the deviation from this point. A method called *equilibrium* approximates the schedule with maximum utility accrual based on an analogy with physical pendulums. In this paper, we propose a scheduling algorithm for this task model to schedule periodic tasks. The basic idea of our solution is to combine the equilibrium with Earliest Deadline First (EDF) in order to reuse EDF's well studied timeliness analysis. We present simulation results and an example multimedia application to show the benefits of our solution.**

I. Introduction

Target sensitive real-time applications have jobs with *target points* at which execution results in highest utility; the utility degrades as a function of the deviation from this point. Scheduling must be able to defer work in order to ensure job execution at the target point, and must account for the arrival of future jobs in order to not cause deadline misses. In case of periodic task scheduling, if phases are 0 and deadlines are less than or equal to the period, the scheduler must consider all jobs within the hyper-period to guarantee timing constraints. Let us define N as the number of jobs, and n as the number of periodic tasks. In a schedule, the number of jobs within the hyper-period can grow in factorial scale with the number of periodic tasks ($N = n!$). Therefore, scheduling algorithms that defer work have a factorial worst case complexity with the number of periodic tasks ($O(n!)$). If phase and deadlines are arbitrary, the number of jobs may be unbounded [1]. On-line scheduling under these conditions is impractical.

One examples of periodic target sensitive application is high media processing, where frames must be displayed strictly periodically and at target points for maximum perceived quality of video. Frames have to be displayed right after decoding e.g. for consumer electronics, and hence, buffering is not an option in this case [2]. If the decoding takes too long and cannot be completed by its deadline, either the decoding has to be aborted or the display delayed. The perceived quality may be higher upon delayed display, provided that the impact on subsequent frames is limited and the final utility clear.

In TUF schedulers, tasks aggregate a given amount of utility to the system as a function of when they complete; the goal of the scheduler is to maximize system utility. A best-effort work-conserving algorithm for resource allocation in computing systems is proposed in [3]. The TUF scheduler proposed in [4] is non-work-conserving, assumes arbitrary types of utility function, and approximates the optimum in $O(N^3)$ under the constraint that all jobs are within the same busy period. In [5], the authors propose a TUF scheduling algorithm with $O(N^2)$ assuming only non-increasing TUFs. The result is used in Ethernet packet scheduling to define the ordering. This work is extended in [6] and [7] to support variable cost functions and mutual exclusion of resources, respectively. In [8] an energy-aware TUF scheduler is proposed, but the focus is to satisfy statistical performance requirements. All these approaches are not capable to defer work to their respective target points (i.e. work-conserving), with exception of the approach proposed in [4]. However, this approach has high on-line overhead (since in the worst case $N = n!$) and imposes a very restrictive limitation: there must be no idle time in the schedule, which is very unlikely to happen.

The gravitational task model [9], [10], [11] uses a method called *equilibrium* to compute the trade-off among the execution of jobs for increased accrued utility. This method has linear complexity w.r.t. the number of jobs, approximates the optimum, and holds provided the order jobs execute is given and jobs have elliptical utility functions. In [10], the authors presented an on-line scheduler with complexity $O(N \times log(N))$ for the gravitational task model that uses the equilibrium and an utility density based heuristic to reorder the execution sequence of jobs for increased utility accrual; scheduling consists of sorting and deferring the execution of jobs. Through the rest of this paper, we will refer to this scheduler as *Grav-DST*. In the case of periodic tasks, in the worst case $N = n!$.

In this paper, we propose a gravitational task model based scheduling algorithm for periodic tasks which is inspired on a mix of EDF and Grav-DST. We call this algorithm *Grav-EDF-swap*. This algorithm uses an interval of time called *equilibrium window* to limit the amount of jobs when computing the equilibrium; we limit the number of jobs to $N = n^2$. We, then, propose a method to compute the equilibrium of jobs which guarantees the timing constraints of jobs outside the equilibrium window. Finally, we propose an heuristic to reorder the execution sequence which increases utility accrual and does not violate timing constraints.

Our apporach significantly reduces the overhead to schedule periodic tasks to $O(N = n^2)$. This lower overhead does not come at the expense of restricted feasibility, and simulation results show that there is a negligible impact on the utility

978-3-9810801-8-6/DATE12/©2012 EDAA

accrual compared to methods that consider full knowledge of the arrival of future jobs.

The rest of this paper is organized as follows. Section II recalls the gravitational task model, and section III describes the Grav-EDF-swap. Section IV brings results from a simulation study. Section V an example multimedia application enhanced with our scheduling algorithm. Finally, section VI concludes the paper.

II. The gravitational task model

The gravitational task model assumes jobs j_i with earliest start time est_i, relative deadline dl_i, worst case execution time $WCET_i$, target point tp_i and importance imp_i; these jobs may be instances of periodic tasks or not. The execution window of a job j_i is, then, the interval $[est_i, est_i + dl_i]$. A job obtains its highest utility at the target point; executing somewhat before or after is feasible, but at lower utility. As the whole execution of a job cannot happen in one point in time, the target point relates to an *anchor point* within the job execution. The value of an anchor point (α_i) is the portion of execution of job j_i that executes before this anchor point ($0 \leq \alpha_i \leq 1$) and d_i is the amount of execution time in between α_i and α_{i+1}. Jobs are not allowed to execute out of their execution window. Each job can express an utility decay as a function of its deviation from its target point. They may or may not be instances of recurring tasks. The importance represents the flexibility of a job to be shifted from its target point in the presence of other jobs. Finally, the utility density is defined as $imp_i/WCET_i$.

Equation (1), called *equilibrium*, calculates the deviation x_n of the last job in a busy period from its target point so that the utility accrual of the whole busy period approximates the maximum. Table I shows the calculation each parameter in this equation using task parameters. Refer to [9], [11] for a complete description of the equilibrium calculation.

$$x_n = \frac{\sum_{i=1}^{n-1} W_i \times (\sum_{j=i}^{n-1}(d_j) + P_i - P_n)}{\sum_{i=1}^{n} W_i} \quad (1)$$

equilibrium parameters	task parameters
W_i	$2 \times imp_i/(dl_i - WCET_i)$
R_i	$(dl_i - WCET_i)/2$
P_i	$est_i + (dl_i - WCET_i)/2 + \alpha_i WCET_i$
d_i	$(1-\alpha_i)WCET_i + \alpha_{i+1}WCET_{i+1}$

TABLE I: Task and equilibrium parameters.

III. The scheduling algorithm

For the sake of simplicity, we split the description of the algorithm into 2 parts: first we describe the algorithm to combine the Gravitational Task Model with EDF (Grav-EDF), and then we extend this algorithm to swap the execution of jobs for increased utility accrual (Grav-EDF-swap).

Algorithm of Grav-EDF. First, let us define an interval of time ew called *equilibrium window*. This interval starts at tc, and ends at ew_end (hence, $ew = [tc, ew\_end[$). In this work, we calculate ew_end so that the equilibrium windows contains n^2 jobs. This restriction in the length of the execution window provides for lower complexity and, as we will see in the evaluation section, has negligible impact on the utility accrual of the schedule.

At system start-up, the scheduler

- calculates ew_end so that n^2 jobs arrive within the interval $[0, ew\_end[$.
- Then, schedules the execution of all jobs that arrive within the equilibrium window as in work-conserving non-preemptive EDF.
- Finally, computes the equilibrium of all jobs under the constraint that no work is deferred in the interval $[ew\_end, \infty[$, i.e. the schedule outside the equilibrium window is as in work-conserving non-preemptive EDF.

During runtime

- Upon completion of a job, the scheduler sets ew_end to the arrival time of the next job outside the current equilibrium window, orders the execution of incoming jobs with the other jobs using EDF, computes the equilibrium of all jobs as described above.

Restricting the equilibrium to delay jobs only within the execution window guarantees that the execution sequence that EDF generates prevails.

The complexity of Grav-EDF is linear with the number of jobs in the equilibrium window, which is the complexity to compute the equilibrium. As we limit the number of jobs within the equilibrium window to n^2, the complexity is $O(n^2)$.

Algorithm of Grav-EDF-swap. Grav-EDF-swap initially applies Grav-EDF as described in section III. Then, it scans all jobs within the equilibrium window, and swaps 2 adjacent jobs if (i) they are in the same busy period and, (ii) upon swap, no timing constraint is violated and (iii) the job with higher utility density lies closer to its target point. In case both jobs have the same utility density, the sum of their absolute deviations must decrease. Only jobs in the same busy period compete for their target points, and hence, are considered for swap. After scanning all jobs, Grav-EDF-swap computes the equilibrium for the jobs once again. The number of times that Grav-EDF-swap scans the jobs for swap is customizable, but simulations results show that more than one round brings little increase in the utility accrual. At runtime, we apply the steps above every time the scheduler admits a new job in the schedule.

Example. Table II contains the parameters of 2 periodic tasks, and table III the first 3 jobs that arrive in the system ($\tau_{i,j}$ is the j^{th} instance of task τ_i). For the sake of simplicity, we assume $ew\_end = 4$. The equilibrium window is $ew = [0, 4[$ at $tc = 0$, and the jobs arriving in this interval are $\tau_{1,1}$, $\tau_{2,1}$, and $\tau_{1,2}$. Scheduling these jobs with EDF results in the schedule depicted in figure 1a. Then, the scheduler computes the equilibrium, resulting in the schedule depicted in figure 1b. Jobs $\tau_{2,1}$ and $\tau_{1,2}$ compete for their target points in this scenario. We apply, then, the reordering heuristic. Jobs $\tau_{2,1}$ and $\tau_{1,2}$ have adjacent executions, and the same utility density — same importance and same execution time. Their deviations

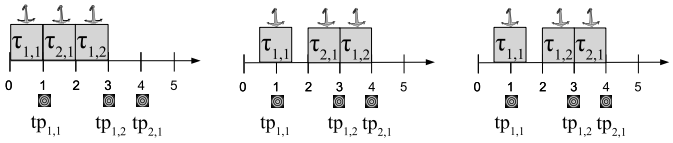
	τ_1	τ_2
EST	0	0
period	2	5
WCET	1	1
relative tp	0.5	0.875
anchor point	0.5	0.5
importance	1	1

TABLE II: Task set.

	$\tau_{1,1}$	$\tau_{1,2}$	$\tau_{2,1}$
EST	0	2	0
deadline	2	4	5
WCET	1	1	1
target point	1	3	4
anchor point	0.5	0.5	0.5
importance	1	1	1

TABLE III: Job set.

before swap are $x(\tau_{2,1}) = -1.5$ and $x(\tau_{1,2}) = 0.5$, and upon swap $x'(\tau_{2,1}) = -0.5$ and $x(\tau_{1,2}) = -0.5$. As upon swap the sum of their absolute deviations from their target points gets smaller, the heuristic swaps them, resulting in the schedule of figure 1c. The scheduler computes, finally, the equilibrium, which does not change the schedule because the completion of $\tau_{2,1}$ may not be deferred beyond the equilibrium window.



(a) *Schedule at* $tc = 0$ *before equilibrium* (b) *Schedule at* $tc = 0$ *after equilibrium* (c) *Schedule at* $tc = 0$ *after equilibrium and swap heuristic*

Fig. 1: Intermediate schedules for Grav-EDF-swap.

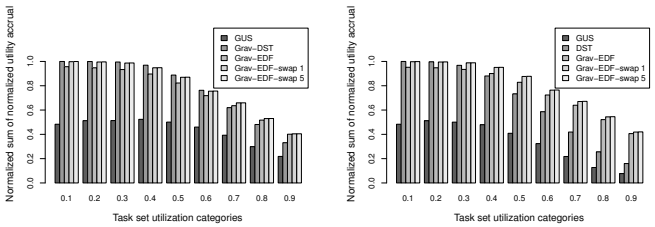
Scanning the job list has complexity $O(N = n^2)$, swapping adjacent jobs has constant complexity, and may happen at most $N - 1$ times (complexity $O(N = n^2)$). Finally, the equilibrium calculation performed at the end has linear complexity. Therefore, the complexity of EDF-swap is $O(N = n^2)$.

IV. Evaluation

In our experiments, we compare Grav-EDF-swap with Grav-DST [10] and the Generic Utility Scheduler (GUS) [7]. GUS is a recent work-conserving TUF scheduler (i.e. does not defer work) and we include it in the comparison due to the non-existence of a non-work-conserving TUF scheduler that can handle idle times in the schedule.

In our simulations, the number of periodic tasks in each task set is a random integer uniformly distributed in the interval $[2, 10]$. The system utilization categories varies in the range $[0.1, 0.9]$ with granularity 0.1. Each utilization category comprises 1000 randomly generated task sets, including infeasible ones. The period and importance of each task are integer numbers uniformly distributed in the interval $[1, 10]$. Deadlines are equal to the period, offsets are 0 and target points vary across experiments. The computation times were uniformly distributed such that the generated task set has the desired utilization. In each schedule, we considered all jobs within the hyper-period. Our results are within a confidence level of 95% with significance level of 0.05.

In the experiment results depicted in figure 2, we compare GUS, Grav-DST, Grav-EDF, Grav-EDF-swap with 1 round of the swap heuristic ('Grav-EDF-swap 1'), and EDF-swap



(a) *Target points equal to* 0.5 (b) *Target points variable*

Fig. 2: Utility accrual for different scheduling algorithms.

with 5 rounds of the swap heuristic ('Grav-EDF-swap 5'). As can be seen, applying one round of the swap heuristic results in $\sim 5\%$ utility increase, while more rounds of the swap heuristic result in negligible increase. Therefore, we conclude that the acceptance ratio of the schedule has more impact on the utility accrual than the actual execution sequence of jobs. Comparing the results for Grav-EDF and Grav-EDF swap, we also conclude that most of the increase in the utility accrual comes from the equilibrium of jobs, and not from the execution sequence of jobs. As expected, GUS performs poorly due to its incapability of deferring work to their respective target points.

We can also observe that 'Grav-EDF-swap 1' is statistically superior or equivalent to Grav-DST in most cases for target point equal to 0.5. Varying the target points does not affect the performance of Grav-EDF-swap, while Grav-DST has up to 50% decrease in utility accrual (compare figures 2a and 2b).

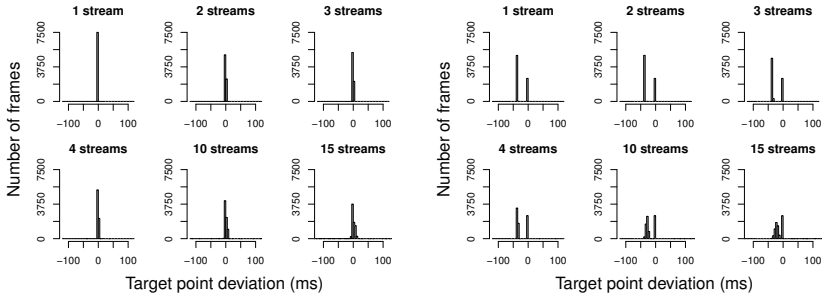
V. Multimedia application

A. Modeling in the gravitational task model

We use Grav-EDF to schedule the decoding and display tasks without the need to buffer extra decoded frames or compromise feasibility; we do not use swap because frames must be decoded in a specific order, and EDF can guarantee this order [2]. The period of each decoding and display task is 1/framerate; the deadline is the length of the GOP (*group of pictures*) so that the high variability of decoding times do not compromise feasibility. The computation time of each decoding task instance is the decoding time of the respective frame (thus, we consider the exact execution time). The decoding task is not target sensitive (only the display), and hence, we set importance 0, which dismisses the need for a target point. Display tasks have execution time 0 (negligible because DMA module handles data transfer between decoding buffer and display buffer), importance of the stream to the user, the target point is the desired display time, and the anchor point is 0 (start of data transfer). The pendulum equilibrium is suitable to schedule the frame display tasks because their utility functions have a concave nature [12].

B. Experiments

We generate periodic task sets from decoding times of 5 MPEG-2 streams originating from DVB-S (*Digital Video Broadcasting—Satellite*). Each stream comprises 7500 interlaced frames with different properties (5 minutes of video in



(a) *Grav-EDF* (b) *EDF*

Fig. 3: Histograms of deviations for car documentary.

each stream), and displays 25 frames per second. The $1st$ stream is a car documentary, and we assign importance 10; the $2nd$ stream is business news, the $3rd$ stream is a volleyball match, and we assign both importance 2; the $4th$ stream is a soccer match, and we assign importance 5; and finally, the $5th$ stream is a cartoon, and we assign importance 2.

We use the measured decoding times to schedule the frame decoding and display using a scheduling simulator—an implementation the scheduling algorithm on a real-time operating system platform can obtain these values from decoding time estimators [13]. Finally, we use the output of the simulator to generate a video output that displays frames according to the schedule of the display tasks.

In our experiments, we use 6 feasible task sets with different workloads. Task set A contains the $1st$ stream; task set B contains the $1st$ and $2nd$ streams; task set C contains the $1st$, $2nd$ and $3rd$ streams; task set D contains the $1st$, $2nd$, $3rd$ and $4th$ streams; task set E contains all streams twice; and finally, task set F contains all streams 3 times. The utilization of those task sets vary in between 10% and 70%.

In figure 3 we plot the histograms for frame display deviation of the car documentary under the Grav-EDF and pure EDF—we omit the results for the other streams due to similarities in all results and space restrictions in this paper. The x-axis is the deviation in milliseconds — each bar of the histogram has a width of 5 milliseconds —, and the y-axis is the number of interlaced frames. Each graph plots the histogram for 6 task sets: A, B, C, D, E, and F. We can see that scheduling the tasks with EDF results in higher dispersion of the deviations for both low and high task set utilizations because some frames are decoded too early. Grav-EDF is able to account for the target point of frames under low workload by deferring the display task, yet providing the necessary flexibility under high workload in order to avoid frame drops (the deviations gradually increase).

Observing the resulting video output, we can see that larger deviations of frame display in EDF result in jerked scenes, while using Grav-EDF results in a smooth video playout.

VI. Conclusion

In this paper, we proposed a gravitational task model based scheduling algorithm for target sensitive periodic tasks. This algorithm, called *Grav-EDF-swap*, has complexity $O(n^2)$; previous scheduling algorithms for the gravitational task model (e.g. [11], [10]) have complexity $O(n!)$. The basic idea is to use an interval of time called *equilibrium window* to limit the amount of jobs considered in the equilibrium computation, and an heuristic to reorder the execution sequence within this window. Simulation results revealed that the low complexity of our scheduling algorithm does not come neither at the expense of a lower acceptance ratio, nor at the expense of a decreased utility accrual. We also presented an multimedia application example to illustrate that target sensitive applications enhanced with our scheduler provide better service quality.

References

[1] S. K. Baruah, R. R. Howell, and L. Rosier, "Algorithms and complexity concerning the preemptive scheduling of periodic, real-time tasks on one processor," *Real-Time Systems*, vol. 2, pp. 301–324, 1990.

[2] D. Isovic, G. Fohler, and L. F. Steffens, "Timing constraints of MPEG-2 decoding for high quality video: misconceptions and realistic assumptions," in *Proceedings of ECRTS 03)*, Porto, Portugal, July 2003.

[3] C. D. Locke, "Best-effort decision-making for real-time scheduling," Ph.D. dissertation, Pittsburgh, PA, USA, 1986.

[4] K. Chen and P. Muhlethaler, "A scheduling algorithm for tasks described by time value function," *Real-Time Syst.*, vol. 10, no. 3, 1996.

[5] J. Wang and B. Ravindran, "Time-utility function-driven switched Ethernet: Packet scheduling algorithm, implementation, and feasibility analysis," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 2, 2004.

[6] H. Wu, U. Balli, B. Ravindran, and E. D. Jensen, "Utility accrual real-time scheduling under variable cost functions," in *Proceedings of RTCSA'05*. Washington, DC, USA: IEEE Computer Society, 2005.

[7] P. Li, H. Wu, S. B. Ravindran, and E. D. Jensen, "A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints," *IEEE Trans. Comput.*, vol. 55, no. 4, 2006.

[8] H. Wu, B. Ravindran, E. D. Jensen, and P. Li, "Energy-efficient, utility accrual scheduling under resource constraints for mobile embedded systems," *Trans. on Embedded Computing Sys.*, vol. 5, no. 3, 2006.

[9] R. Guerra and G. Fohler, "A gravitational task model for target sensitive real-time applications," in *ECRTS08 - 20th Euromicro Conference on Real-Time Systems*, Prague, Czech Republic, July 2008.

[10] ——, "On-line scheduling algorithm for the gravitational task model," in *ECRTS09 - 21th Euromicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009.

[11] ——, "A gravitational task model with arbitrary anchor points for target sensitive real-time applications," *Real-Time Syst.*, vol. 43, no. 1, 2009.

[12] M. Claypool and J. Tanner, "The effects of jitter on the perceptual quality of video," *Proc. ACM Multimedia '99*, 1999.

[13] J. Hamers and L. Eeckhout, "Resource prediction for media stream decoding," in *Proceedings of the conference on Design, automation and test in Europe*, ser. DATE '07. San Jose, CA, USA: EDA, 2007.