

# An Optimum Generalized Equilibrium Solution for the Gravitational Task Model\*

Raphael Guerra and Gerhard Fohler  
Chair for Real Time Systems  
Technische Universität Kaiserslautern, Germany  
{guerra,fohler}@eit.uni-kl.de

## Abstract

*In some applications, each job has a preferred point in time for execution, which we call target point, but can shift around this point, albeit at lower utility. We call these applications target sensitive. As scheduling all jobs directly at their target points might result in overlapping executions, the trade-off that maximizes the accrued utility of the system without violating timing constraints must be addressed. In [7] we proposed the gravitational task model, which is able to express the target demand of such applications and compute an approximation of their trade-off assuming elliptical utility functions. This task model is based on a physical pendulum analogy, which makes the understanding of the problem and solution intuitive. Nonetheless, target demands may differ among applications and even among jobs within the same application.*

*In this paper, we address these issues with a novel method to find the trade-offs which is not bound to a particular function and holds even if each job has a different type of utility function. This method generalizes the method in [7] and is optimum. Moreover, it maintains the intuition of the physical pendulum analogy.*

## 1 Introduction

Real-time systems are traditionally defined as computing systems that must react to events within time constraints to provide correct behavior [3]. Therefore, most real-time schedulers were developed having deadlines as primary concern. This timeliness criteria is used to express an interval of time (*execution window*) where the task is allowed to execute. As a result, there is the assumption that the utility of tasks is constant within their execution window [13].

Some applications have *target sensitive constraints*: each job should preferably execute at a specific point within its execution window, called *target point*, but can

execute around this point, albeit at lower utility. The utility of a job is, hence, defined as a function of the deviation from the target point. The utility function of jobs may vary depending on applications' properties, differing even among jobs within the same application. As executing all jobs directly at their target points might not be feasible due to overlapping executions, the trade-off among the schedule of their executions must be computed so that the accrued utility of the system is maximized.

Examples for such applications include media processing and control. In media processing, frames must be displayed at target points for the best user perceived quality of the video stream. Buffering frames in advance is not an option in these applications [10], hence frames must be displayed before the next frame starts being decoded. If frames cannot be displayed at their target points, e.g., due to the execution of other jobs, they can be either dropped or delayed. The impact of dropping/delaying the display of a frame on the perceived quality of video may vary among frames [10]. In control, sampling and actuation are ideally executed at their target points for optimum output [15]. Shifting them a little bit for the sake of feasibility is acceptable provided the system response remains within bounded limits. On the other hand, jitter in the computations executed between samplings and actuations do not jeopardize the system. As can be seen, the sensitivity to deviations from the target points may vary among applications and even among jobs within the same application.

Time utility function (TUF) scheduling as presented in [5, 18, 13] and earliness/tardiness schedulers [2] go beyond the starttime-deadline notion to express tasks' temporal constraints. In TUF schedulers, tasks aggregate a given amount of utility to the system as a function of when they execute; the goal of the scheduler is to maximize system utility. A study on several types of time utility functions was presented in [11]. The work in [16] proposes a utility function based approach to express the importance of tasks in order to assign resources. This work covers multiple resources and investigates how utility functions can be used to solve the resource allocation problem, but presents no scheduling algorithm. A best-effort solution for resource allocation in computing systems was pro-

\*This work has been partially supported by the EU FP-7 project ACTORS (<http://www.actors-project.eu/>).

posed in [14]. However, scheduling decisions are based only on the utility that tasks accrue to the system at the current point in time and not on the shape of their utility functions over time. A TUF scheduler assuming any kind of utility function was proposed in [5]. It uses a heuristic to find an ordering on average close to the optimum in  $O(n^3)$ , under the constraint that work cannot be deferred. In [18], the same problem was solved in  $O(n^2)$  assuming only non-increasing TUFs. The result was used in ethernet packet scheduling, to define the ordering. Since the utility decreases upon delayed arrival, packets are sent as early as possible after being ordered. This work was extended in [19] and [12] to support variable cost functions and mutual exclusion of resources, respectively. In [20] an energy-aware TUF scheduler was proposed, but the focus is to satisfy statistical performance requirements.

Earliness/tardiness schedulers seek the minimization of the overall penalty of the system due to tasks that complete too early or too late. However, the penalty function is linear with the distance from the deadline and without bounds. In this approach deadlines are allowed to be missed, which is not part of our assumptions. The computational time and quality of such schedulers depend mainly on the ordering algorithms and not on the temporal constraint. A small survey on earliness/tardiness schedulers presented in [2] shows that many are based on branch and bound and most of them can only schedule tasks in a busy period.

In classic task models the utility of jobs cannot be expressed, but target sensitivity can be enforced by tightening the offsets and deadlines, at the expense of decreasing the maximum feasible utilization. A method to decrease the variation in the completion time of tasks under EDF was proposed in [1]. This problem was solved by decreasing the deadlines of tasks based on their importance, provided that with the new deadline assignment the schedule is still feasible. Although this work states the possibility of adjusting offsets as well, this approach does not take offsets into account. Therefore, target points within the execution windows cannot be addressed.

The elastic scheduling of [4] presents a method that improves the acceptance of tasks under overload situations in EDF. In order to accept a task, it increases the period of tasks to reduce the utilization of the system based on a spring system analogy. This work does not consider the target sensitivity when readjusting periods.

In [6] we proposed a gravitational task model for target sensitive applications which allows jobs to express target points and utilities. This task model is based on a physical pendulum analogy which makes the understanding of the problem and solution intuitive. Furthermore, we presented a method with linear complexity to compute the trade-off among the execution of jobs for improved accrued utility which is based on an analogy with the equilibrium of pendulums. We called this method *equilibrium*. This equilibrium approximates the optimum and holds provided the order jobs execute is given and jobs have the same type of

utility functions, which must be elliptical. We extended this work in [7] to allow jobs to express an arbitrary point with their execution to relate to the target points. In [8] we presented an on-line scheduler for the gravitational task model that uses this equilibrium. Finally, we presented in [9] a video adaptation strategy which uses the gravitational task model to compute the trade-off between delaying and dropping the display of a frame for final improved quality of video and resource usage. However, this work assumed that the impact of delay on frame display does not vary among frames, as the gravitational task model was not able to handle jobs with different utility functions.

In this paper, we present a generalization of the equilibrium of the gravitational task model [7] which we call *generic equilibrium*. It holds not only for elliptical, but for any continuously differentiable concave utility function. It holds also for sets of jobs which have different types of concave utility functions. A closed-form equation does not exist for the general case; the generic equilibrium consists of a uni-dimensional equation that needs to be calculated for each case. Therefore, we present an example of an analytical solution to the generic equilibrium for a particular concave and continuously differentiable function. As in some cases a closed-form solution cannot be provided, we prove that the generic equilibrium can always be solved using any numerical root-finding algorithm. Finally, we demonstrate how the generic equilibrium can be used to schedule jobs with an scheduling example.

The generic equilibrium is optimum and covers a large set of utility functions, yet it keeps the intuition resulting from the physical pendulum analogy. It also allows applications' jobs to have mixed types of utility functions in the schedule. Therefore, particular needs of each target sensitive application can be individually expressed (e.g. the variability among different frames of the impact of delay on frame display on the user perceived quality of video).

As a final contribution, we compare the generic equilibrium and the equilibrium proposed in [7]. First, we show that the equilibrium, besides being an approximation of the trade-off for elliptical utility functions, is optimum for a particular class of quadratic utility functions. Then, we empirically compare the approximation proposed in [7] with the numerical solution of the generic equilibrium.

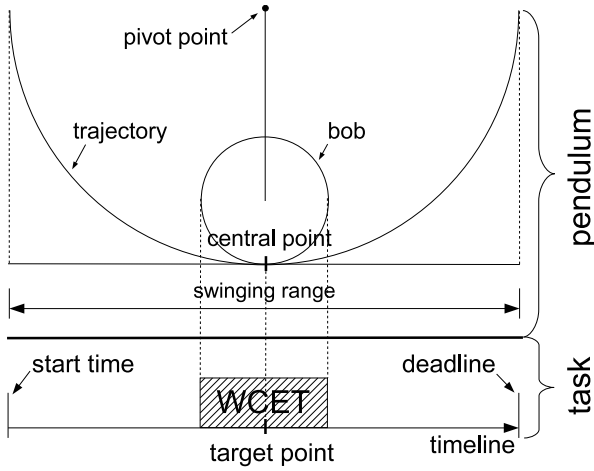
The rest of this paper is organized as follows: Section 2 recalls the gravitational task model presented in [7] as needed for this paper; Section 3 describes the generic equilibrium and Section 4 brings an example of how it can be solved analytically; Section 5 describes a scheduling example which is designed to exploit the generic equilibrium; Section 6 brings results from a simulation study; and, finally, Section 7 concludes the paper.

## 2 The gravitational task model

In this section, we briefly recall the gravitational task model [7] as needed for the rest of this paper. We start

describing a pendulum system and drawing the analogies with real time systems. Then, we formally describe the gravitational task model.

A pendulum is an object attached to a pivot point that can swing freely. A basic example is the simple gravity pendulum or bob pendulum. As depicted in Figure 1, it consists of a bob at the end of a massless string, which, when given an initial push, will swing back and forth under the influence of gravity over its central (lowest) point in a circular trajectory. Placed at the lowest point, the bob will come to rest there (*rest position*). If the bob pendulum contains more than one bob, they cannot be all at the same time in the lowest part, and hence, will push each other aside to find a new rest position. It is said that the system is then in an *equilibrium state*. The *equilibrium condition* is that the sum of all torques in the system is equal to zero and the distance between the centers of two consecutive bobs is the sum of their radii. How far from the central point each bob comes to rest in the equilibrium state depends on their weights and sizes.



**Figure 1. Analogy between bob pendulum and task.**

Drawing the analogy, we can think of a bob as a job whose execution time is equivalent to the size of the bob. A job is allowed to execute at its target point in the absence of other jobs in the system with the same target point. The target point is equivalent, thus, to the central (lowest) point of a pendulum trajectory and the swinging range is the execution window of the job. The importance of a job, which represents its resistance to be shifted away from its target point when interacting with other jobs, can be seen as the weight of the bob. The heavier a bob is, the closer to the bottom it will come to rest. Finally, the job utility as a function of its deviation from the target point is similar to the potential energy of a bob as a function of its deviation from the central point. As the equilibrium is the state that minimizes the potential energy of the pendulum, the best compromise of the job's interests maximizes the accrued utility of the system. This analogy is depicted in figure 1

and summarized in table 1.

pendulum	task set
bob	job
weight	importance
swinging range	execution window
central point	target point
potential energy function	utility function
equilibrium state	best compromise

**Table 1. Analogy between bob pendulum and task set**

The gravitational task model (depicted in figure 5) assumes non-preemptive jobs  $j_i$  in a uni-processor system. They have earliest start time  $est_i$ , relative deadline  $dl_i$  (absolute deadline  $abs\_dl_i = est_i + dl_i$ ), worst case execution time  $WCET_i$ , target point  $tp_i$  and importance  $imp_i$ . They may or may not be instances of recurring tasks. Since the entire execution of a job cannot be performed at a single point in time, we select a point  $\alpha_i$  within its execution, called *anchor point*, which is used to relate a job to its target point. We define the value of  $\alpha_i$  as the fraction of  $WCET_i$  executed before the anchor point, which ranges between 0 and 1, where 0 corresponds to the beginning and 1 the end. Thus, the distance  $d_i$  between two consecutive anchor points is  $(1 - \alpha_i) \times WCET_i + \alpha_{i+1} \times WCET_{i+1}$ . The *execution window* of a job is defined as the range where the anchor point can be placed without violating the timing constraints. Jobs are not allowed to execute outside their execution window. The variation of the *utility*  $g_i$  of a job within its execution window as a function of its deviation  $x_i$  from the target point is given by equation 1. Refer to [7] for a detailed description of the gravitational task model.

$$g_i(x_i) = \frac{2 \times imp_i}{dl_i - WCET_i} \times \sqrt{\left(\frac{dl_i - WCET_i}{2}\right)^2 - x_i^2} \quad (1)$$

Although the analogy between bob pendulums and real time task scheduling appears straightforward, a number of issues have to be addressed before a direct mapping. See [7] for the detailed discussion and motivation. In the final analogy depicted in figure 5 we consider the anchor point of each job as a particle that must be kept at least a horizontal distance  $d_i$  from the next particle, has a weight  $W_i$  and hangs on a pivot point  $P_i$  by a massless string of length  $R_i$ .

In the new analogy,  $d_i$  is the same as in the gravitational task model,  $R_i$  is equal to half the length of the execution window of a job,  $P_i$  is equal to the target point  $tp_i$  and  $W_i = 2 \times imp_i / (dl_i - WCET_i)$ . All pivot points are aligned perpendicularly to the gravity, i.e. they are at the same height. This consideration is important because we can use the same reference point to compute the potential

energy of the particles in the equilibrium state, which is equivalent to the utility accrued by a job in the analogy.

A set of jobs in a busy period is called a *job chain*  $c_k$  in the gravitational task model. It contains jobs that would have overlapping execution if placed at their target points (see figure 5) and is equivalent to a set of particles that push each other aside in a particle pendulum. In [7], we derived an approximation of the trade-off in a job chain with  $N$  jobs that maximizes the accrued utility inspired on the equilibrium condition of physical pendulums. It is given by equation 2, which is called *equilibrium* and computes the deviation of the last job in the chain from its target point. The equilibrium holds provided the order jobs execute is given and all jobs have the same type of utility function, which must be elliptical. A detailed description of the equilibrium can be found in [7]; here we focus on results, as needed for the work in this paper.

$$x_N = \frac{\sum_{i=1}^{N-1} W_i \times (\sum_{j=i}^{N-1} (d_j) + tp_i - tp_N)}{\sum_{i=1}^N W_i} \quad (2)$$

### 3 Generic equilibrium

The trade-off in a job chain of  $N$  jobs maximizing the accrued utility of the system for a given order of jobs is expressed by the constrained maximization problem 3. The goal function to be maximized is the sum of the utilities of each job, subject to the constraints that there is no idle time between consecutive jobs (1st constraint in (3)) and that the anchor points are not placed outside the respective execution windows (2nd and 3rd constraints in (3)). This mathematical formulation is equivalent to the equilibrium condition of pendulums. In this section, we describe how to convert this constrained multi-dimensional maximization problem into one single unconstrained uni-dimensional maximization problem which has an optimum solution. Unlike the equilibrium equation derived directly from the pendulum analogy, this solution holds assuming jobs with any continuously differentiable concave utility function  $g_i$ . The utility function  $g_i$  of each job can be of different types, e.g. elliptical, parabolic, hyperbolic, etc.

$$\max : g(x_1, \dots, x_n) = \sum_{i=1}^N g_i(x_i)$$

s.t :

$$\begin{aligned} tp_{i+1} + x_{i+1} - (tp_i + x_i) &= d_i & \forall i = 1..N-1 \\ x_i &\leq abs\_dl_i - (1 - \alpha_i) \times WCET_i - tp_i & \forall i = 1..N \\ x_i &\geq est_i + \alpha_i \times WCET_i - tp_i & \forall i = 1..N \end{aligned} \quad (3)$$

The goal function  $g$  is an  $N$  dimensional problem as function of  $x_1, \dots, x_n$ . However, the 1st constraint can be rewritten to express  $x_i$  as a function of  $x_N$ , hence reducing the problem to 1 dimension. The steps that lead to equation 4 show this transformation.

$$\begin{aligned} tp_{i+1} + x_{i+1} - (tp_i + x_i) &= d_i \\ \sum_{j=i}^{N-1} (tp_{j+1} + x_{j+1} - (tp_j + x_j)) &= \sum_{j=i}^{N-1} d_j \\ tp_N + x_N - (tp_i + x_i) &= \sum_{j=i}^{N-1} d_j \end{aligned}$$

finally:

$$x_i(x_N) = tp_N - tp_i - \sum_{j=i}^{N-1} d_j + x_N \quad (4)$$

As a result,  $g(x_1, \dots, x_n)$  becomes a sum of composite functions of  $g_i$  and  $x_i$  as a function of  $x_N$ , as can be seen in equation 5.

$$g(x_N) = \sum_{i=1}^N (g_i(x_i(x_N))) = \sum_{i=1}^N (g_i \circ x_i)(x_N) \quad (5)$$

By replacing equation 4 in the 2nd and 3rd constraints, which describe the space where  $g(x_1, \dots, x_n)$  is defined, we obtain inequality system 6.

$$\begin{cases} x_N \leq abs\_dl_i - (1 - \alpha_i) \times WCET_i - tp_N + \sum_{j=i}^{N-1} d_j \\ x_N \geq est_i + \alpha_i \times WCET_i - tp_N + \sum_{j=i}^{N-1} d_j \\ \forall i = 1..N \end{cases} \quad (6)$$

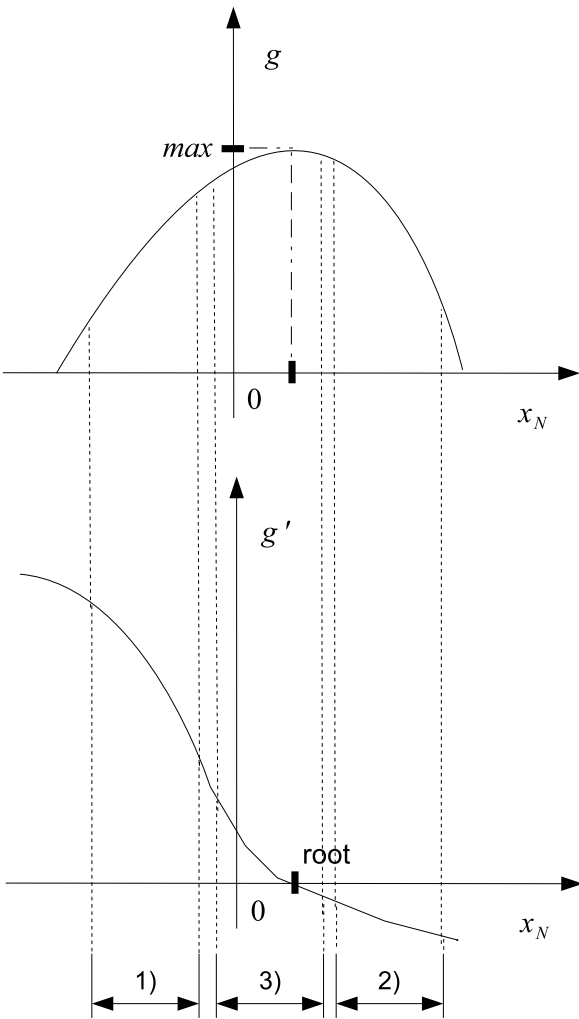
Solving this inequality system gives the interval  $I$  where equation 5 is defined. If  $I$  is empty, there exists no trade-off that meets the timing constraints of all jobs at the same time. Otherwise, the schedule that maximizes the accrued utility in a job chain is given by  $x_N \in I$  that maximizes equation 5. Once  $x_N$  is found, the deviations of the other jobs from their target points are calculated using equation 4.

Next we show how to find the maximum value of  $g(x_N)$  in any interval  $I$  provided  $g(x_N)$  is concave. The sum of continuously differentiable concave functions is also a continuously differentiable concave function [17]. Therefore, if each job  $j_i$  has a concave and continuously differentiable utility function  $g_i(x_i)$ , then  $g(x_N)$  is concave and continuously differentiable in interval  $I$ . We assume applications provide utility functions to the system along with their closed-form derivatives.

A function is called concave in an interval iff its derivative is monotonically decreasing in this interval [17]. Figure 2 shows a concave function and its derivative. If  $g(x_N)$  is concave and continuously differentiable,  $g'(x_N)$  has at most one root in any interval  $I$ , where  $I$  must fall in one of the 3 categories below (depicted in figure 2):

1. the interval contains only positive values of  $g'(x_N)$ , thus  $g(x_N)$  is monotonically increasing with maximum at the rightmost point of  $I$ .

2. the interval contains only negative values of  $g'(x_N)$ , thus  $g(x_N)$  is monotonically decreasing with maximum at the leftmost point of  $I$ .
3. the interval contains both positive and negative values of  $g'(x_N)$ , thus  $g(x_N)$  has maximum at  $g'(x_N) = 0$ , which is the absolute maximum.



**Figure 2. Example of a concave function and its derivative.**

We differentiate  $g(x_N)$  by applying the chain rule to solve  $\sum_{i=1}^N (g_i \circ x_i)'(x_N)$ , thus obtaining equation 7, which we call *generic equilibrium*. Note that  $x_i'(x_N) = 1, \forall i = 1..N$ . If  $I$  falls in either case 1) or 2), finding  $x_N$  that maximizes the accrued utility is straightforward. Otherwise, we must find the root of the generic equilibrium.

$$g'(x_N) = 0$$

$$\sum_{i=1}^N (g_i \circ x_i)'(x_N) = 0$$

$$\sum_{i=1}^N (g_i' \circ x_i)(x_N) \times x_i'(x_N) = 0$$

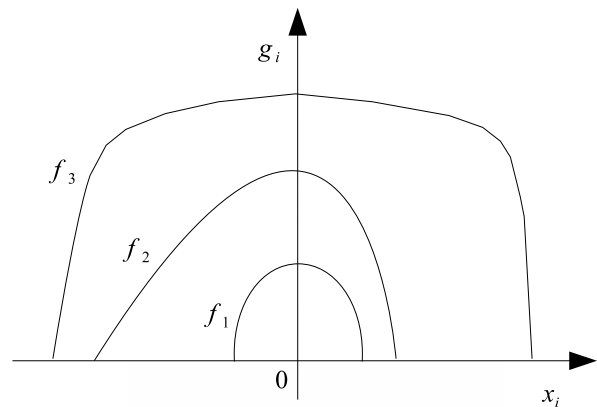
finally:

$$\sum_{i=1}^N (g_i' \circ x_i)(x_N) = 0 \quad (7)$$

The root of the generic equilibrium can be found either analytically or using any numerical (iterative) root-finding algorithm [17], e.g. the bisection method, Newton's method, etc. A discussion on these methods is beyond the scope of this paper. An analytical solution is preferable, but it is not always possible.

Solving equation 7 consists of applying the numerical method rule to the sum of  $N$  elements ( $\sum_{i=1}^N (g_i' \circ x_i)(x_N)$ ) until the return value of this sum converges to 0. Each iteration of the numerical root-finding algorithms has complexity  $O(1)$  and  $\sum_{i=1}^N (g_i' \circ x_i)(x_N)$  is computed with complexity  $O(N)$  as in algorithm 1.

Figure 3 shows some examples of types of continuously differentiable concave utility functions. Note that since the gravitational task model defines that the utility of jobs is maximized at the target points,  $g_i(x_i)$  is maximized when the deviation from the target point is zero, i.e.  $x_i = 0$ . We assume that a negative deviation happens to the left side of the target point and a positive to the right.



**Figure 3. Some continuous differentiable concave utility functions.**

The complexity to solve the generic equilibrium is  $O(K * N)$ , where  $K$  denotes the number of iterations necessary to converge and is independent of the number of jobs. Note that the efficiency of the solution also depends on the rate of convergence of the chosen root-

---

**Algorithm 1** Computing  $\sum_{i=1}^N (g'_i \circ x_i)(x_N)$ .

---

```

sum_d = - d[N]
result = 0
i = N
while i > 0
{
  sum_d = sum_d + d[i]
  x[i] = tp[N] - tp[i] - sum_d + x[N]
  result = result + g'[i](x[i])
  i = i - 1
}
return result

```

---

finding algorithm. Solving  $g'_i[x[i]]$  analytically is independent of any other job rather than  $j_i$ , hence having complexity  $O(1)$ . Therefore, an analytical solution of the generic equilibrium has complexity  $O(N)$ . The next section brings an example of an analytical solution of the generic equilibrium.

#### 4 Example of analytical solution of the generic equilibrium

The generic equilibrium presented in section 3 holds for jobs with any continuously differentiable concave utility function. We proposed a solution using root-finding algorithms because a closed-form solution does not exist in the general case. In this section, we present an example of an analytical solution of the generic equilibrium assuming quadratic utility functions as in equation 8.

$$g_i(x_i) = a_i x_i^2 + b_i x_i + c_i \quad (8)$$

Figure 4 depicts a valid quadratic utility function. As  $g_i(x_i)$  must be concave, we have  $a < 0$ . Furthermore,  $g_i(x_i)$  must be maximized at  $x_i = 0$  to be a valid utility function, and hence, we have  $g'_i(0) = 0$ . As can be seen in equation 9,  $g'_i(0) = 0$  only if  $b_i = 0$ . At  $x_i = 0$  the return value is  $c_i$ .

$$g'_i(x_i) = 2a_i x_i + b_i \quad (9)$$

Replacing equation 9 in the generic equilibrium (equation 7), we obtain equation 10 after a few steps for  $b_i = 0$ . The outer sum in the numerator can be reduced to range from 1 to  $N - 1$  because, for  $i = N$ ,  $tp_i = tp_N$  and  $\sum_{j=i}^{N-1} d_j = 0$ , which zeros out the term.

Note that for  $a_i = W_i$  equation 10 becomes equation 2, which is the equilibrium proposed in [7]. Therefore, besides being an approximation of the trade-off for elliptical utility functions, this equilibrium is optimum for a particular class of quadratic utility functions. This class of quadratic utility functions has  $a_i = (2 \times imp_i)/(dl_i - WCET_i)$ ,  $b_i = 0$  and  $c_i = imp_i$ .

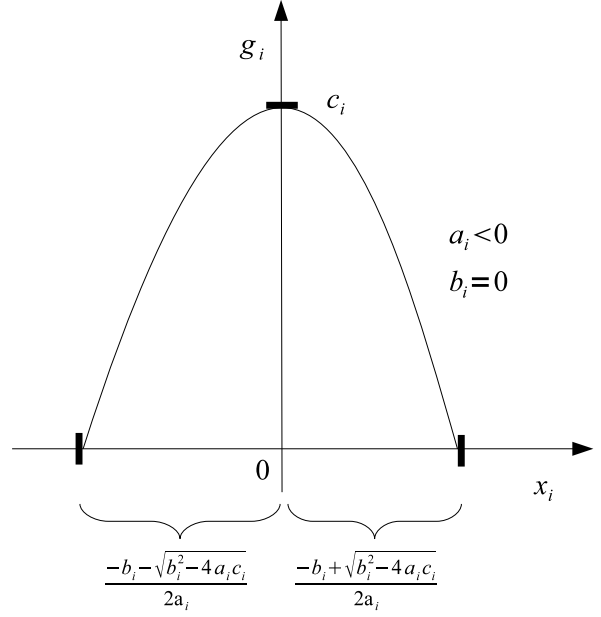


Figure 4. A quadratic utility function.

$$\begin{aligned}
\sum_{i=1}^N (g'_i \circ x_i)(x_N) &= 0 \\
\sum_{i=1}^N 2a_i x_i(x_N) &= 0 \\
\sum_{i=1}^N \left( 2a_i \times \left( tp_N - tp_i - \sum_{j=i}^{N-1} d_j + x_N \right) \right) &= 0 \\
\sum_{i=1}^N 2a_i x_N + \sum_{i=1}^N 2a_i \times \left( tp_N - tp_i - \sum_{j=i}^{N-1} d_j \right) &= 0
\end{aligned}$$

finally:

$$x_N = \frac{\sum_{i=1}^{N-1} a_i (\sum_{j=i}^{N-1} d_j) + tp_i - tp_N}{\sum_{i=1}^N a_i} \quad (10)$$

#### 5 Scheduling example

In this section, we present an example of a scheduling algorithm that uses the generic equilibrium. This algorithm is divided into 2 intertwined phases: *ordering phase* and *timing phase*. In the *ordering phase* the scheduling algorithm decides the order jobs will execute. In [6, 7] the authors proposed a static ordering by target points, and in [8] the authors proposed a dynamic ordering heuristic based on the utility density of jobs for increased utility accrual. For the sake of simplicity, we use the static ordering proposed in [6, 7]. In the so-called *timing phase*, after the insertion of each job in the schedule, the scheduler uses the generic equilibrium to compute the deviation of each

job from its target point that results in maximum accrued utility for the particular ordering. This timing phase with the generic equilibrium is fully compliant with the ordering phase proposed in previous work.

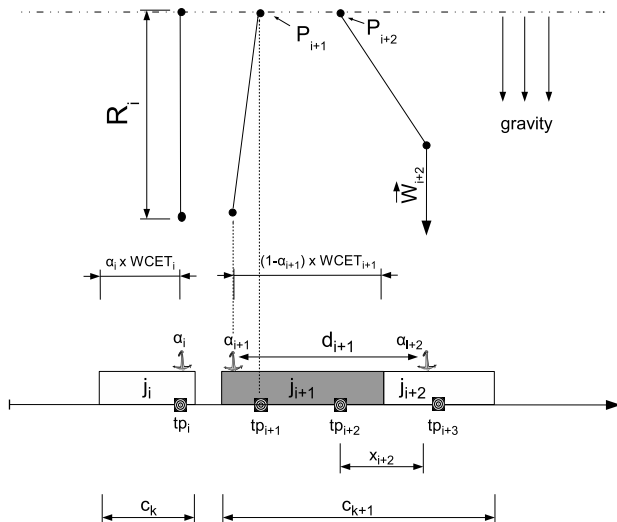


Figure 5. Scheduling example 1.

Assume the scheduling of  $M$  jobs (see figures 5 and 6). First of all, job chains must be detected in order to apply the generic equilibrium. Therefore, system jobs are inserted into the schedule one by one and, upon overlapping execution with a job chain in the schedule, the inserted job becomes part of this chain and the generic equilibrium is computed as described in section 3. In the case depicted in figure 6, job  $j_{i+3}$  becomes a part of job chain  $c_{k+1}$  after being inserted into the schedule depicted in figure 5. Note that the generic equilibrium shifts the new chain, which might merge with other chains, creating a ripple effect. In figure 6 we observe that job chains  $c_k$  and  $c_{k+1}$  present in figure 5 merge into a single chain. At each such merge the generic equilibrium must be computed for the new chain. As can be seen in the figures, this situation is similar to adding a new particle in a pendulum; the particles will push each other aside and come to rest in a new equilibrium state where new interactions with particles might happen. An inserted job which does not cause any execution overlap is considered as a new job chain in the schedule (as job  $j_i$  in figure 5). Once all jobs have been inserted into the schedule, all job chains are known and their schedule maximizes the accrued utility of the system for the given execution order.

This scheduling algorithm presented here serves only as example, and is not the focus of this paper. This algorithm has complexity  $O(K * N^2)$ , since the scheduling algorithm applies the timing phase  $N$  times (for each job insertion). In order to apply this algorithm on-line, arrival of future jobs may have to be taken into account in the schedule because the timing phase delays the execution of jobs. Scheduling recurring tasks also deserves special treatment. In case tasks are strictly periodic and dead-

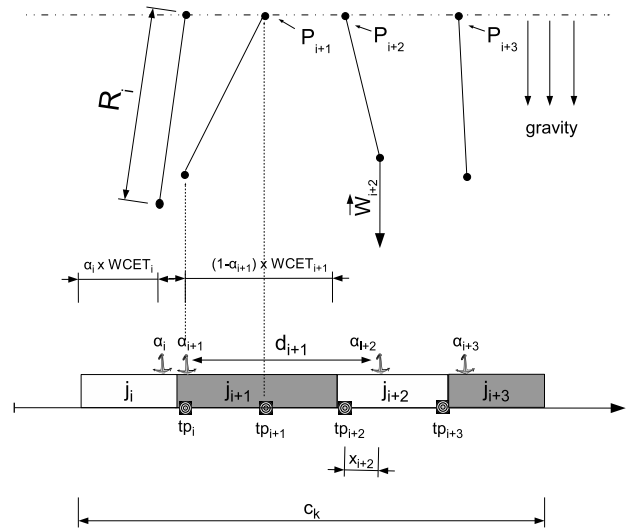


Figure 6. Scheduling example 2.

lines are not larger than the periods, it suffices to schedule all jobs within the hyper-period at the beginning of each hyper-period. Further aspects of the development of scheduling algorithms for the gravitational task model can be found in [8].

## 6 Evaluation

In our simulations, task sets comprise periodic tasks and the system utilization varies in the range  $[0.1, 0.9]$  with granularity 0.1. Each utilization category comprises 1000 randomly generated task sets, including infeasible ones. We do not discard infeasible task sets because they are also a property of the scheduling algorithm. As we consider only feasible task sets for evaluation of the utility accrual, the actual number of task sets per utilization category (in increasing order of utilization) are 1000, 1000, 991, 916, 748, 603, 472, 339, and 237. The evaluation results for each utilization category are within a confidence level of 95% with significance level of 0.05.

The number of periodic tasks in each task set is a random integer uniformly distributed in the interval  $[2, 10]$ . The period and importance of each task are integer numbers uniformly distributed in the interval  $[1, 10]$ . Deadlines are equal to the period, earliest start times are equal to 0 and target points lie in the middle of the execution window of the jobs, thus allowing the same amount of deviation to both sides of the target point. 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 hyperperiod and ordered them by their target points.

We consider 5 different types of continuously differentiable utility functions (where  $R_i = \frac{dl_i - WCET_i}{2}$ ):

- $g1_i(x_i) = imp_i \times \sqrt{1 - \left(\frac{x_i}{R_i}\right)^2}$

- $g2_i(x_i) = imp_i \times \sqrt{1 - \left(\frac{x_i}{R_i}\right)^4}$
- $g3_i(x_i) = imp_i \times \left(1 - \left(\frac{x_i}{R_i}\right)^4\right)$
- $g4_i(x_i) = imp_i \times \left(2 - \frac{e^{\frac{1.31695 \times x_i}{R_i}} + e^{-\frac{1.31695 \times x_i}{R_i}}}{2}\right)$
- $g5_i(x_i) = imp_i \times \left(1 - \left(\frac{x_i}{R_i}\right)^2\right)$

The respective derivatives are:

- $g1'_i(x_i) = -\frac{imp_i \times \left(\frac{x_i}{R_i}\right)}{R_i \times \sqrt{1 - \left(\frac{x_i}{R_i}\right)^2}}$
- $g2'_i(x_i) = -2 \times \frac{imp_i \times \left(\frac{x_i}{R_i}\right)^3}{R_i \times \sqrt{1 - \left(\frac{x_i}{R_i}\right)^4}}$
- $g3'_i(x_i) = -4 \times imp_i \times \left(\frac{x_i}{R_i}\right)^3 \times \frac{1}{R_i}$
- $g4'_i(x_i) = -imp_i \times \frac{1.31695}{R_i} \times \left(\frac{e^{\frac{1.31695 \times x_i}{R_i}} - e^{-\frac{1.31695 \times x_i}{R_i}}}{2}\right)$
- $g5'_i(x_i) = -2 \times imp_i \times \left(\frac{x_i}{R_i^2}\right)$

The utility function  $g1_i(x_i)$  is the same as in equation 1 after a few algebraic steps to convert from the pendulum domain to the task set domain (see section 2), and  $g5_i(x_i)$  is a quadratic polynomial function as in equation 8 (see section 4). The other utility functions are arbitrary.

In our experiments, we compare the generic equilibrium with the equilibrium proposed in [7]. Other TUF schedulers are not applicable because they do not consider the target sensitivity of tasks to delay work, i.e. those schedulers are work-conserving. The metric of comparison is the accrued utility of each task set; acceptance ratio is left out of this comparison because we are interested in evaluating the generic equilibrium, rather than the example scheduling algorithm. Therefore, we calculate for each feasible task set the error of the accrued utility obtained by the original equilibrium, which is  $1 - \frac{g^{eq}}{g^{opt}}$  ( $g^{eq}$  stands for the utility accrued by the original equilibrium and  $g^{opt}$  stands for the utility accrued by the generic equilibrium, which is optimum).

In the experiment depicted in figure 7, all jobs have utility function  $g1_i(x_i)$ . To the best of our knowledge, a closed-form of the generic equilibrium for elliptical functions is not possible; we solve the the generic equilibrium using the bisection method [17]. This figure contains the cumulative distribution functions (C.D.F.) of the error of  $g^{eq}$ ; the y-axis is in polynomial scale to the power 40. Task sets are categorized by their utilization in the ranges [0.1, 0.3], [0.4, 0.6] and [0.7, 0.9]. Under low system utilization jobs rarely overlap execution when scheduled at

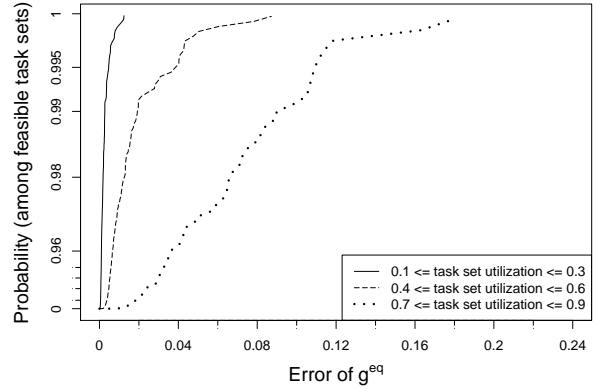


Figure 7. C.D.F. of error of  $g^{eq}$  (jobs with utility function  $g1_i(x_i)$ ).

their target points. Hence, we can see that for task set utilization within [0.1, 0.3] the equilibrium and the generic equilibrium have very close results. As the utilization increases, the approximation of the trade-off among jobs' executions loses accuracy. The line for task set utilization within [0.7, 0.9] shows that the error in the original equilibrium is smaller than 4% in approximately 96% of the cases, but in 4% of the cases this error ranges from 4% to approximately 18%. The superiority of the generic equilibrium does not come at the expense of a higher complexity.

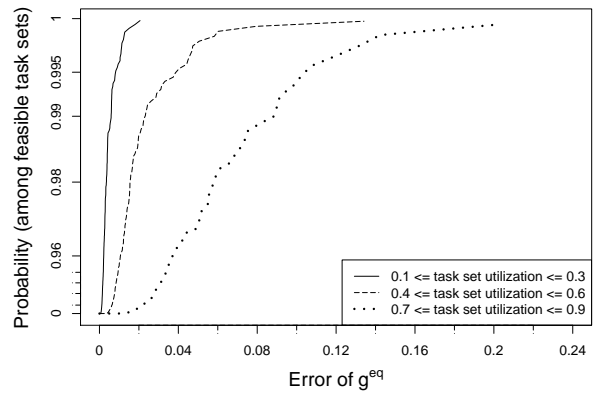


Figure 8. C.D.F. of error of  $g^{eq}$  (jobs with either utility function  $g1_i(x_i)$ ,  $g2_i(x_i)$ ,  $g3_i(x_i)$ ,  $g4_i(x_i)$  or  $g5_i(x_i)$ ).

In the experiment depicted in figure 8, each job has one of the 5 utility functions listed earlier. All other job parameters remain as in the previous experiment. Once again we observe that for task set utilization within [0.1, 0.3] the original equilibrium and the generic equilibrium have very

close results and that the error goes up to 20% in some cases for high task set utilization. In fact, the experiments depicted in figures 7 and 8 have very similar results.

## 7 Conclusion

In some applications, each job has a preferred point in time for execution, called *target point*, but can shift around this point, albeit at lower utility. We call these applications *target sensitive*. The utility function of jobs may vary depending on applications' properties, differing even among jobs within the same application. As scheduling all jobs directly at their target points might result in overlapping executions, the trade-off that maximizes accrued utility of the system without violating timing constraints must be addressed.

In this paper we have introduced a new method to compute this trade-off generalizing the gravitational task model we proposed earlier, which is based on a intuitive physical pendulum analogy. There, we proposed an approximation inspired by such pendulums called *equilibrium*. It assumes a given order of job executions with all jobs having the same type - elliptical - of utility functions. The *generic equilibrium* method proposed here is a generalization of the equilibrium presented earlier, in that it is not bound to a particular utility function and holds even if jobs of the same application have different utility functions. These features address the particular needs of target sensitive applications such as control and media processing. The generic equilibrium is optimum and holds for any continuously differentiable concave utility function, yet keeps the intuition resulting from the physical pendulum analogy.

We presented an example of a scheduling algorithm which uses the generic equilibrium. Simulation results showed the superiority of this scheduler in accruing utility over a scheduler using the equilibrium presented earlier.

On-going and future work include explicit consideration of full preemption, determining job ordering and support for variable job parameters (e.g. importances, execution times, etc.).

## 8 Acknowledgment

The authors wish to express their gratitude to the members of the real-time systems group at TU Kaiserslautern for the good discussions and reviewing of the paper, notably Jens Theis and Stefan Schorr. The authors acknowledge furthermore the comments of the reviewers of this paper.

## References

- [1] S. Baruah, G. Buttazzo, S. Gorinsky, and G. Lipari. Scheduling periodic task systems to minimize output jitter. In *RTCSA '99: Proceedings of the Sixth International Conference on Real-Time Computing Systems and Applications*, page 62, Washington, DC, USA, 1999. IEEE Computer Society.
- [2] K. Bülbül, P. Kaminsky, and C. Yano. Preemption in single machine earliness/tardiness scheduling. *J. of Scheduling*, 10(4-5):271–292, 2007.
- [3] G. C. Buttazzo. *Hard Real-time Computing Systems: Predictable Scheduling Algorithms And Applications (Real-Time Systems Series)*. Springer-Verlag TELOS, Santa Clara, CA, USA, 2004.
- [4] G. C. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computer*, 51(3):289–302, 2002.
- [5] K. Chen and P. Muhlethaler. A scheduling algorithm for tasks described by time value function. *Real-Time Syst.*, 10(3):293–312, 1996.
- [6] 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.
- [7] R. Guerra and G. Fohler. A gravitational task model with arbitrary anchor points for target sensitive real-time applications. *Real-Time Syst.*, 43(1):93–115, 2009.
- [8] R. Guerra and G. Fohler. On-line scheduling algorithm for the gravitational task model. In *ECRTS09 - 21th Euromicro Conference on Real-Time Systems*, Dublin, Ireland, July 2009.
- [9] R. Guerra and G. Fohler. Video decoding for reduced jitter and improved resource usage. In *WRT09: XI Workshop de Tempo Real*, Recife, PE, Brazil, May 2009.
- [10] D. Isovich, G. Fohler, and L. F. Steffens. Timing constraints of MPEG-2 decoding for high quality video: misconceptions and realistic assumptions. In *15th Euromicro Conference on Real-time Systems (ECRTS 03)*, Porto, Portugal, July 2003.
- [11] E. D. Jensen. Asynchronous decentralized real-time computer systems. In W. A. Halang and A. D. Stoyenko, editors, *Real-Time Computing*, the NATO Advanced Study Institute. Springer Verlag, October 1992.
- [12] P. Li. A utility accrual scheduling algorithm for real-time activities with mutual exclusion resource constraints. *IEEE Trans. Comput.*, 55(4):454–469, 2006. Member-Haisang Wu and Senior Member-Binoy Ravindran and Member-E. Douglas Jensen.
- [13] P. Li, B. Ravindran, and E. D. Jensen. Adaptive time-critical resource management using time/utility functions: Past, present, and future. In *COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference - Workshops and Fast Abstracts - (COMPSAC'04)*, pages 12–13, Washington, DC, USA, 2004. IEEE Computer Society.
- [14] C. D. Locke. *Best-effort decision-making for real-time scheduling*. PhD thesis, Pittsburgh, PA, USA, 1986.
- [15] P. Marti, G. Fohler, K. Ramamritham, and J. M. Fuertes. Control performance of flexible timing constraints for quality-of-control scheduling. In *Proceedings of the 23rd IEEE Real-Time Systems Symposium*, Texas, TX, USA, Dec. 2002.
- [16] D. Prasad, A. Burns, and M. Atkins. The valid use of utility in adaptive real-time systems. *Real-Time Syst.*, 25(2-3):277–296, 2003.
- [17] W. Sun and Y.-x. Yuan. *Optimization Theory and Methods*, volume 1. springer, 2006.

- [18] J. Wang and B. Ravindran. Time-utility function-driven switched Ethernet: Packet scheduling algorithm, implementation, and feasibility analysis. *IEEE Trans. Parallel Distrib. Syst.*, 15(2):119–133, 2004.
- [19] H. Wu, U. Balli, B. Ravindran, and E. D. Jensen. Utility accrual real-time scheduling under variable cost functions. In *RTCSA '05: Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05)*, pages 213–219, Washington, DC, USA, 2005. IEEE Computer Society.
- [20] 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.*, 5(3):513–542, 2006.