Optimal Elastic Scheduling

Xiaobo Sharon Hu and Thidapat Chantem
Department of Computer Sci & Engr
University of Notre Dame
Notre Dame, IN 46556
{shu,tchantem}@cse.nd.edu

M.D. Lemmon
Department of Electrical Engineering
University of Notre Dame
Notre Dame, IN 46556
lemmon@nd.edu

Abstract

This paper introduces an optimization framework for the elastic scheduling of periodic tasks. The paper rederives the original elastic scheduling algorithm in [2] as the solution to an optimization problem that seeks to minimize the squared deviation of a task's utilization from initial desired utilization. We apply this approach to develop an elastic scheduling algorithm that seeks to minimize the average difference of task period from a desired minimum period subject to various inequality constraints.

1. Introduction

A desirable property of a real-time system is the guarantee that it will perform at least beyond some pre-specified thresholds defined by system designers. This is usually not a concern under normal situations where analysis has previously been done offline to ensure performance of the system based on its regular workload. However, in response to an event such as user's input or changing environment, the load of the system may dynamically change in such a way that a temporal overload condition occurs. The challenge, then, is to provide some mechanisms to guarantee the minimum performance level under such circumstances.

Many scheduling algorithms (e.g. [3]) have been proposed that allow some jobs to be dropped to improve schedulability. However, it is sometimes more suitable to execute tasks less often instead of dropping them. For example, limitations on the throughput capacity of ad hoc communication networks [1] make it highly desirable to reduce overall network traffic by having control tasks adaptively adjust their periods in response to the actual activity level of the control application. To address this type of requirements, [2] proposed a more flexible framework known as the elastic task model where deadline misses are avoided by increasing tasks periods until some desirable utilization level is achieved.

We will focus our attention on said elastic task model. The framework is based on an elegant analogy between spring systems and task scheduling in which a task's *resistance* to changing its period is viewed as a spring's resistance to being compressed. In accordance with the "principle of least action" found in classical mechanics, this suggests that the elastic task model is really attempting to minimize some overall measure of the task set's "energy", whose precise nature was not made clear in the original work. This paper re-examines that question and identifies the specific measure of task set energy whose minimization leads to the task compression algorithm [2]. We feel that identifying such a cost function is important because it provides guidance in the selection of weighting factors, such as elastic coefficients from the elastic task model, and because it suggests that it may be possible to generalize the original elastic task framework in which elastic coefficients are chosen to minimize other relevant measures of task set energy.

The remainder of the paper is organized as follows. In Section 2, we present our approach to period selection. Section 3 shows our experimental results and the paper concludes with Section 4.

2 Formal Approach to Period Selection

Given a particular set of real-time tasks, there may exist numerous sets of periods for the tasks to be schedulable. It is not difficult to see that different sets of periods would lead to different performance of the resultant system. In general, the period selection problem can be thought of as an optimization problem. That is,

```
optimize:    performance metric
subject to:  tasks are schedulable
             bounds on periods are satisfied
```

Below we introduce two specific performance metrics and discuss their implications. We assume that tasks are

scheduled according to the EDF scheduling policy and task deadlines equal to task periods.

2.1 Minimize utilization perturbation

Processor utilization by each task is an important measure for any real-time system. It not only reveals the amount of system resource dedicated to the task but also impacts schedulability. In the elastic task model, one consequence of changing task periods is changing the utilization of tasks. From the stand point of performance preservation, it is desirable to minimize the changes in task utilization. This objective can be captured by the following constrained optimization problem.

$$\text{minimize:} \qquad E(U_1, \cdots, U_N) = \sum_{i=1}^{N} w_i \left(U_{i0} - U_i\right)^2 \quad (1)$$

$$\text{subject to:} \qquad \sum_{i=1}^{N} U_i \leq U_d \quad (2)$$

$$U_i \geq U_{i\min} \quad \text{for } i = 1, 2, \cdots, N \quad (3)$$

In the formulation, N is the number of tasks in the system, U_{i0} is the initial utilization of task τ_i and $U_{i0} \geq U_{i\min}$, U_i is the new utilization of τ_i to be determined, and U_d is the desired total utilization. (U_d is usually set to 1 for EDF scheduling.) Constant w_i (≥ 0) is a weighting factor and reflects the criticality of a task. More critical tasks would have larger w_i's. Thus, w_i can be considered as the *value* of task τ_i. The first constraint simply states the schedulability condition under EDF. The rest of the constraints bound the utilization, equivalently bound the task periods by T_{imax} where $U_{i\min} = C_i/T_{imax}$.

The above constrained optimization problem belongs to the category of quadratic programs and can be solved in polynomial time. However, solving such a problem during runtime can be too costly. What makes the above formulation attractive is that the solution to the above problem is exactly the same as that found by the task compression algorithm presented in [2]. We introduce several lemmas and a theorem to support this argument. Due to the page limit, we omit some of the proofs.

Lemma 1 *Given the constrained optimization problem as specified in (1)-(3), any solution, U_i^*, to the problem must satisfy $\sum_{i=1}^{N} U_i^* = U_d$.*

Proof: *This is proven through a straightforward application of the Kuhn-Tucker necessary conditions.* □

Lemma 2 *Given the constrained optimization problem as specified in (1)-(3), any solution, U_i^*, to the problem must satisfy*

$$U_i^* = U_{i0} - \frac{\frac{1}{w_i}\left(\sum_{U_j^* \neq U_{jmin}} U_{i0} - U_d + \sum_{U_j^* = U_{jmin}} U_j\right)}{\sum_{U_j^* \neq U_{jmin}} (1/w_j)} \quad (4)$$

if $U_i^* > U_{imin}$, and $U_i^* = U_{imin}$ otherwise.

Proof: According to the Kuhn-Tucker conditions, the necessary conditions for the existence of a relative minimum at U_i^* are

$$0 = \frac{\partial J_a}{\partial U_i^*} = -2w_i \left(U_{i0} - U_i^*\right) + \mu_0 - \mu_i, \quad i = 1, \cdots, N \quad (5)$$

$$0 = \mu_0 \left(U_d - \sum_{i=1}^{N} U_i^*\right) \quad (6)$$

$$0 = \mu_i \left(U_{imin} - U_i^*\right) \quad i = 1, \cdots, N \quad (7)$$

where μ_i's are Lagrange multipliers and $\mu_i \geq 0$ for $i = 0, \cdots, N$. From Lemma 1, we know that any solution to the given problem must satisfy (6), i.e.,

$$U_d = \sum_{i=1}^{N} U_i^*. \quad (8)$$

Furthermore, some other constraints in (7) may also be active. Suppose the k-th constraint in (7) is active, we have $U_i^* = U_{imin}$. Then, from

$$\mu_k = \mu_0 - 2w_i \left(U_{k0} - U_{kmin}\right) \quad (9)$$

We can then solve for μ_0 by summing up Equation (5) for all i and obtain

$$\mu_0 = \frac{2\left(\sum_{U_i^* \neq U_{imin}} U_{i0} - U_d + \sum_{U_i^* = U_{imin}} U_{imin}\right)}{\sum_{U_i^* \neq U_{imin}} (1/w_i)}$$

If $\sum_{U_i^* \neq U_{imin}} U_{i0} + \sum_{U_i^* = U_{imin}} U_{imin} > U_d$, it is easy to verify that $\mu_0 > 0$ and $\mu_k \geq 0$. Therefore, U_i^* is a solution to the optimization problem. □

Readers can readily verify that the periods obtained by the algorithm in [2] satisfies the necessary conditions given in Lemma 1 and 2. However, there may be more than one solution that satisfies the conditions in Lemma 1 and 2. To determine a globally optimal solution, a brute force method is to examine all combinations of U_i^* that satisfy the necessary conditions. What needs to be shown is that the periods found by the elastic algorithm in [2] indeed minimizes the objective function in (1). The following lemma is indispensable in arriving at the above conclusion.

Lemma 3 *Given the constrained optimization problem as specified in (1)-(3), let* $\sum_{i=1}^{N} U_i' = U_d$ *and*

$$U_i'' = U_{i0} - \frac{\frac{1}{w_i}\left(\sum_{U_j' \neq U_{jmin}} U_{i0} - U_d + \sum_{U_j'=U_{jmin}} U_{jmin}\right)}{\sum_{U_j' \neq U_{jmin}} (1/w_j)}$$

If $U_k'' < U_{kmin}$ *for some* $1 \leq k \leq N$*, then*

$$E(\overline{U}_1, \ldots, \overline{U}_k, \ldots, \overline{U}_N) \leq E(\widehat{U}_1, \ldots, \widehat{U}_k, \ldots, \widehat{U}_N)$$

where $\overline{U}_i$ *and* $\widehat{U}_i$ *both satisfy the necessary conditions in Lemma 1 and 2, and* $\overline{U}_k = U_{kmin}$ *and* $\widehat{U}_k > U_{kmin}$*.*

Based on the above lemma and setting $w_i = 1/e_i$ (where e_i is the elastic coefficient used in [2]), the following theorem can be readily proved.

Theorem 1 *Consider a task set of* N *tasks where* U_i *is the utilization of the* i*th task. Let* U_{i0} *denote the initial desired utilization of task* τ_i *and let* $e_i > 0$ *be a set of* elastic coefficients *for* $i = 1, \ldots, N$*. Let* $U_0 = \sum_{i=1}^{N} U_{i0}$*. Task utilization* U_i *obtained from the task compression algorithm in [2] minimizes*

$$E(U_1, \ldots, U_N) = \sum_{i=1}^{N} \frac{1}{e_i}(U_{i0} - U_i)^2$$

subject to the inequality constraints $\sum_{i=1}^{N} U_i \leq U_0$ *and* $U_i \geq U_{imin}$ *(*$i = 1, \ldots, N$*).*

The above theorem has two significant consequences. First, it reveals the optimization criterion inherent in the task compression algorithm. Secondly, it illustrates that the task compression algorithm (with a time complexity of $O(n)$)) can be used to solve certain convex programming problems.

2.2 Minimize task periods

For some applications, instead of focusing on utilization, it may be more important to examine task periods directly. For example, in a control application, task periods reflect the sampling periods of certain control tasks. The performance of a control system is very much dependent on the sample periods used. When dynamic overload occurs and some task periods must be adjusted to ensure schedulability, it can be more desirable to satisfy schedulability while minimizing task periods.

If the range of possible period values for each task is unbounded, the following constrained optimization problem can be used to minimize the changes in task periods.

$$\text{minimize:} \qquad J(T_1, \cdots, T_N) = \sum_{i=1}^{N} w_i(T_i - T_{i0}) \quad (10)$$

$$\text{subject to:} \qquad \sum_{i=1}^{N} \frac{C_i}{T_i} \leq 1 \quad (11)$$

In the formulation, N is the number of tasks in the system, C_i is the execution time of task τ_i, T_{i0} is its initial period, and T_i is its new period to be determined. As before, $w_i \geq 0$ is a weighting factor that reflects the criticality of a task. The constraint in equation (11) is simply the usual condition assuring the task set is schedulable under EDF. (Other utilization bounds can be readily used in (11).) Let T_i^* denote a locally optimal set of task periods for the above optimization problem. The following theorem provides a closed-form characterization of this minimizer.

Theorem 2 *Given the constrained optimization problem specified in equations (10)–(11), a locally optimal solution is*

$$T_i^* = \sqrt{\frac{C_i}{w_i}} \sum_{k=1}^{N} \sqrt{w_k C_k} \quad (12)$$

Proof: This theorem is proved through a straightforward application of the Kuhn-Tucker necessary conditions. □

Remark: If we require that $T_i \geq T_{i_{min}}$ (in other words a lower bound on the desired period), then the above equation becomes a set of inequality constraints, i.e.,

$$\sqrt{w_i} T_{i_{min}} \leq \sqrt{C_i} \sum_{k=1}^{N} \sqrt{w_k} \sqrt{C_k}$$

which can be used to identify a set of admissible weighting factors whose selection ensures $T_i \geq T_{i_{min}}$.

3. Experimental Results

This section uses an example to illustrate the application of our flexible scheduling framework. The results in this paper will verify the equivalence of the flexible scheduling algorithm in [2] and the optimal solution given in theorem 1. This section's results will also evaluate the performance of flexible scheduling as specified in theorem 2 in which we try to minimize average perturbations in task period. We reused the first task set provided in the experimental results section of [2], which is reproduced below in Table 1. The task compression algorithm was written in `C++`, while the results obtained from the constrained optimization problems and the closed-form expressions were obtained from `MatLab`.

Table 1. Task set parameters used

Task	C_i	T_{i0}	$T_{i_{min}}$	$T_{i_{max}}$	e_i
τ_1	24	100	30	500	1
τ_2	24	100	30	500	1
τ_3	24	100	30	500	1.5
τ_4	24	100	30	500	2

In this experiment, all tasks start at time 0 with an initial period of 100 time units. The required minimum utilization

of the overall system is $\frac{24}{500}+\frac{24}{500}+\frac{24}{500}+\frac{24}{500} = 0.192$. Since the current utilization is $\frac{24}{100} + \frac{24}{100} + \frac{24}{100} + \frac{24}{100} = 0.96$, the task set is schedulable under EDF. When, at time 10000, τ_1 wants to change its period to 33 time units, it is allowed to do so since the new required minimum utilization of the system is $\frac{24}{33} + \frac{24}{500} + \frac{24}{500} + \frac{24}{500} = 0.871$, which is less than 1. However, since the EDF scheduling algorithm is assumed, τ_2, τ_3, and τ_4 can no longer execute with their initial period, as the current utilization of the system is $\frac{24}{33} + \frac{24}{100} + \frac{24}{100} + \frac{24}{100} = 1.45$. In other words, the period of tasks τ_2, τ_3, and τ_4 must increase for the system to remain schedulable. Figures 1 shows the resulting period of τ_2 when applying the task compression algorithm, the constrained optimization problem on minimum utilization perturbation, as well as the constrained optimization problems on minimum unbounded period and upper bounded period, along with their respective closed-form expressions. Note that only the results for τ_2 are shown due to space limit.

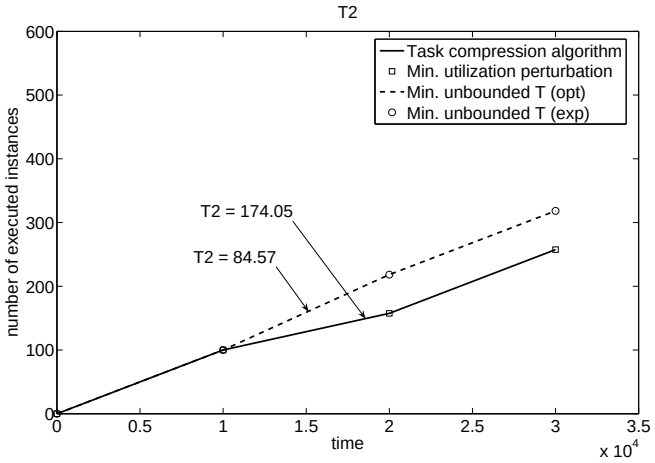

Figure 1. Optimal periods for τ_2

Figures 1 shows how τ_2 changes its period based on the different mechanisms used. The number of executed instances is plotted as a function of time unit. It can easily be seen from the graph that the task compression algorithm [2] and the optimal solution from theorem 1 are the same. The constrained optimization problem on minimum unbounded period and its corresponding closed-form expression seem to be equivalent (in the figure, the result of the constrained optimization problem is denoted by "opt" and that of the closed-form expression by "exp"). In addition, it is worth noting that the value given by each constrained optimization problem is different from the others. However, T_2 is optimal in each case with respect to the corresponding objective function, as well as constraints.

It should be clear by now that the results from the constrained optimization problems are not comparable to each other and that there is no "best" result for all situations. System designers must make the decision of selecting the most useful objective function given a specific system and its constraints.

4. Conclusion and Future Work

As some important questions–such as how one can systematically select weighting factors (e.g. elastic coefficients) or what performance metric the task compression algorithm seeks to minimize–were left unanswered when the elastic task model was introduce, we attempted to address these questions by creating a more general framework where the task compression algorithm can be treated as a special case. This paper represents an effort to view tradeoffs as optimization problems with possibly infinite performance criterion. Once system designers select an appropriate goal for their systems, they can readily use this framework to find a corresponding set of optimal values. Properly formulating a problem can make it easier to find an efficient algorithm or expression that will optimize a specific performance measure. Ultimately, our framework allows for flexible task scheduling where an agreement between control system designers and real-time system designers can be reached.

As future work, we plan on exploring different objective functions, such as those that involve the frequency or utilization of a task, using different types of cost functions in which some may not necessarily result in closed-form expressions.

Since the case for aperiodic tasks may impose different constraints, such as response time, it is also worth investigating. Lastly, this framework can also be used to treat cases where the task deadline is less than its period or where the task computation time is variable.

References

[1] P. Antsaklis and J. B. (editors). Special issue on networked control systems. *IEEE Transactions on Automatic Control*, 49:1421–1423, September 2004.

[2] G. Buttazzo, G. Lipari, M. Caccamo, and L. Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, March 2002.

[3] M. Hamdaoui and P. Ramanathan. A dynamic priority assignment technique for streams with (m,k)-firm deadlines. *IEEE Transactions on Computers*, December 1995.