Real-Time Systems manuscript No.
(will be inserted by the editor)

Linear-Time Admission Control for Elastic Scheduling

Received: date / Accepted: date

Abstract Prior algorithms that have been proposed for the uniprocessor implementation of systems of elastic tasks have computational complexity quadratic ($O(n^2)$) in the number of tasks n, for both initialization and for admitting new tasks during run-time. We present a more efficient implementation in which initialization takes quasilinear ($O(n \log n)$), and on-line admission control, linear ($O(n)$), time.

Keywords Preemptive uniprocessor scheduling · Elastic tasks · Admission control

§**1. Introduction.** The elastic recurrent real-time workload model [1,2] provides a framework for dealing with overload by compressing (i.e., reducing) the effective utilizations of individual tasks until the cumulative utilization falls below the utilization bound that can be accommodated. Each task $\tau_i = (U_i^{\min}, U_i^{\max}, E_i)$ is characterized by the minimum amount of utilization $U_i^{\min}$ that it must be provided and the maximum amount $U_i^{\max}$ that it is able to use, as well as an additional elasticity parameter E_i that "*specifies the flexibility of the task to vary its utilization*" [1]. Given a system $\Gamma = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n such elastic tasks, the objective is to assign each task τ_i a utilization U_i, $U_i^{\min} \leq U_i \leq U_i^{\max}$, such that (1) $\sum_{i=1}^{n} U_i$ is as large as possible but bounded from above by a specified constant U_d which denotes the maximum cumulative utilization that can be accommodated; and (2) if $U_i > U_i^{\min}$ and $U_j > U_j^{\min}$ then U_i and U_j must satisfy the relationship [1]

$$\left(\frac{U_i^{\max} - U_i}{E_i}\right) = \left(\frac{U_j^{\max} - U_j}{E_j}\right) \tag{1}$$

A task system Γ for which such U_i exist for all the tasks is said to be feasible. An algorithm was presented in [1, Fig. 3] for determining feasibility and of computing the appropriate values for the utilizations —the U_i's— of feasible systems in $O(n^2)$ time. Essentially this same algorithm was also repurposed in [1] for admission control: for determining whether a new task seeking to join an already-executing system

[1] For tasks τ_i having $E_i = 0$, $U_i = U_i^{\min}$, and therefore the relationship needs not be satisfied.

could be admitted without compromising feasibility, and if so, recomputing the utilization values for the new task as well as for all preëxisting ones. Extensions to elastic scheduling that were proposed by Chantem et al. [3,4] reformulate the problem of determining the utilizations as a quadratic programming problem. This allows the iterative technique in [1] to be applied to a more general class of problems. However, this reformulation continues to have quadratic time-complexity. In this short note we present a more efficient implementation of the algorithm of [1, Fig. 3] that determines feasibility and computes the U_i values in $O(n \log n)$ time, and does admission control in $O(n)$ time.

§**2. Overview of Prior Results.** Let Γ denote a feasible task system with $E_i > 0$ for all tasks[2] $\tau_i \in \Gamma$, and consider the U_i values that bear witness to this feasibility (i.e., each U_i either equals $U_i^{\min}$, or satisfies Expression 1). The tasks in Γ may be partitioned into two classes Γ_{VARIABLE} (those tasks for which $U_i > U_i^{\min}$, and which can therefore have their utilizations "varied" –compressed– further if necessary) and Γ_{FIXED} (those for which $U_i = U_i^{\min}$; i.e., their utilizations are now "fixed"). It has been shown [1, Eqn (8)] that for each $\tau_i \in \Gamma_{\text{VARIABLE}}$

$$U_i = U_i^{\max} - \left(\frac{U_{\text{SUM}} - (U_d - \Delta)}{E_{\text{SUM}}} \right) \times E_i \tag{2}$$

where $U_{\text{SUM}} = \left(\sum_{\tau_i \in \Gamma_{\text{VARIABLE}}} U_i^{\max} \right)$ and $E_{\text{SUM}} = \left(\sum_{\tau_i \in \Gamma_{\text{VARIABLE}}} E_i \right)$ respectively denote the sum of the $U_i^{\max}$ parameters and the E_i parameters of all the tasks in Γ_{VARIABLE}, and $\Delta = \left(\sum_{\tau_i \in \Gamma_{\text{FIXED}}} U_i^{\min} \right)$ denotes the sum of the $U_i^{\min}$ parameters of all the tasks in Γ_{FIXED}.[3] Given a set of elastic tasks Γ, the algorithm of [1, Fig. 3] starts out computing U_i values for the tasks assuming that they are all in Γ_{VARIABLE} — i.e., their U_i values are computed according to Expression 2. If any U_i so computed is observed to be smaller than the corresponding $U_i^{\min}$ then that task is moved from Γ_{VARIABLE} to Γ_{FIXED}, the values of U_{SUM}, E_{SUM}, and Δ are updated to reflect this transfer, and U_i values recomputed for all the tasks. The process terminates if no computed U_i value is observed to be smaller than the corresponding $U_i^{\min}$. It is easily seen that one such iteration (i.e., computing U_i values for all the tasks) takes $O(n)$ time. Since an iteration is followed by another only if some task is moved from Γ_{VARIABLE} to Γ_{FIXED} and there are n tasks, the number of iterations is bounded from above by n. The overall running time for the algorithm of [1, Fig. 3] is therefore $O(n^2)$.

§**3. Our Approach.** Let us define an attribute ϕ_i for elastic task τ_i as follows:

$$\phi_i \stackrel{\text{def}}{=} \left(\frac{U_i^{\max} - U_i^{\min}}{E_i} \right) \tag{3}$$

[2] All tasks τ_i with $E_i = 0$ must have $U_i \leftarrow U_i^{\max}$ in order to satisfy Expression 1; we assume this is done in a pre-processing step, and the value of U_d updated to reflect the remaining available utilization.

[3] Observe that Δ equals the amount of utilization that is allocated to the tasks in Γ_{FIXED}; therefore $(U_d - \Delta)$ represents the amount available for the tasks in Γ_{VARIABLE}, and $\left(U_{\text{SUM}} - (U_d - \Delta)\right)$ the amount by which the cumulative utilizations of these tasks must be reduced from their desired maximums. As shown in the RHS of Expression 2, under elastic scheduling this reduction is shared amongst the tasks in proportion to their elasticity parameters: τ_i's share is (E_i / E_{SUM}).

Algorithm 1: Elastic_Compression(Γ, U_d)

Input: A list Γ of elastic tasks sorted in non-decreasing order of their ϕ_i parameters (see Expression 3) and a desired utilization U_d
Output: Feasibility and the list Γ with computed U_i values

$U_{\text{SUM}} = 0$; $E_{\text{SUM}} = 0$; $\Delta = 0$
forall $\tau_i \in \Gamma$ **do**
    $U_{\text{SUM}} = U_{\text{SUM}} + U_i^{\max}$
    $E_{\text{SUM}} = E_{\text{SUM}} + E_i$
end
forall $\tau_i \in \Gamma$ **do**
    if $\left(U_i^{\max} - \frac{U_{\text{SUM}} - (U_d - \Delta)}{E_{\text{SUM}}} \times E_i \leq U_i^{\min}\right)$ **then**
        //Task τ_i is no longer compressible – it's in Γ_{FIXED}
        $U_i = U_i^{\min}$ //Since $\tau_i \in \Gamma_{\text{FIXED}}$
        $\Delta = \Delta + U_i^{\min}$ //This additional amount of utilization is allocated to tasks in Γ_{FIXED}
        if $(\Delta > U_d)$ **then return** INFEASIBLE;
        //Cannot accommodate the minimum requirements
        $U_{\text{SUM}} = U_{\text{SUM}} - U_i^{\max}$ //Since τ_i is removed from Γ_{VARIABLE}
        $E_{\text{SUM}} = E_{\text{SUM}} - E_i$ //As above — since τ_i is removed from Γ_{VARIABLE}
        $i = i + 1$ //Proceed to considering the next task...
    else
        //Remaining tasks are all compressible (i.e., in Γ_{VARIABLE})
        $U_i = U_i^{\max} - \frac{U_{\text{SUM}} - (U_d - \Delta)}{E_{\text{SUM}}} \times E_i$ // As per Expression 2
    end
end
return FEASIBLE

We will prove a result (Theorem 1 below) that allows us to conclude that in the algorithm of [1, Fig. 3], ***tasks may be "moved" from Γ_{VARIABLE} to Γ_{FIXED} in order of their ϕ_i parameters***.

Assuming that the tasks are indexed in a linked list such that $\phi_i \leq \phi_{i+1}$ for all $i, 1 \leq i < n$, we can then simply make a *single* pass through all the tasks from τ_1 to τ_n, identifying, and computing U_i values for, all the ones in Γ_{FIXED} before any of the ones in Γ_{VARIABLE}. With appropriate book-keeping (see the pseudo-code in Algorithm 1) this can all be done in a single pass in $O(n)$ time. The cost of sorting the tasks in order to arrange them according to non-increasing ϕ_i parameters is $O(n \log n)$, and hence dominates the overall run-time complexity: determining feasibility and computing the U_i parameters can be done in $O(n \log n) + O(n) = O(n \log n)$ time.

Admission control – determining whether it is safe to add a new task and re-computing all the U_i parameters if so – requires that the new task be inserted at the appropriate location in the already sorted list of preëxisting tasks — this can be achieved in $O(n)$ time. Once this is done, the U_i values can be recomputed in $O(n)$ time by the pseudo-code in Algorithm 1. Similarly, removing a task from the system and recomputing the U_i values also takes $O(n)$ time since sorting is not needed.

§**4. A Technical Result.** We now present the main technical result in this short note.

Theorem 1 *If $\tau_i \in \Gamma_{\text{FIXED}}$ and $\phi_i \geq \phi_j$ then $\tau_j \in \Gamma_{\text{FIXED}}$.*

Proof Consider some iteration of the algorithm of [1, Fig. 3] such that τ_i and τ_j both start out in Γ_{VARIABLE}, but τ_i is determined to belong in Γ_{FIXED} in this iteration. This implies that $U_i^{\min}$ is at least as large as the value of U_i that is computed according to Expression 2:

$$U_i^{\min} \geq U_i^{\max} - \left(\frac{U_{\text{SUM}} - (U_d - \Delta)}{E_{\text{SUM}}}\right) \times E_i$$

By algebraic simplification of the above, we have

$$\left(\frac{U_{\text{SUM}} - (U_d - \Delta)}{E_{\text{SUM}}}\right) \geq \left(\frac{U_i^{\max} - U_i^{\min}}{E_i}\right) \tag{4}$$

Note that the LHS of Expression 4 does not contain any term specific to τ_i and so is the same for all the tasks in Γ_{VARIABLE} for this iteration, and that the RHS is simply ϕ_i. Since $\phi_i \geq \phi_j$ (as per the statement of the theorem), we may conclude by the transitivity of the $\geq$ operator on the real numbers that the LHS of Expression 4 would also be $\geq \phi_j$; equivalently, the value of $U_j^{\min}$ is no smaller than the value of U_j that is computed according to Expression 2, and as a consequence τ_j, too, should be moved to Γ_{FIXED} . □

References

1. Giorgio C. Buttazzo, Giuseppe Lipari, and Luca Abeni. Elastic task model for adaptive rate control. In *IEEE Real-Time Systems Symposium*, 1998.
2. Giorgio C. Buttazzo, Giuseppe Lipari, Marco Caccamo, and Luca Abeni. Elastic scheduling for flexible workload management. *IEEE Transactions on Computers*, 51(3):289–302, March 2002.
3. T. Chantem, X. S. Hu, and M. D. Lemmon. Generalized elastic scheduling. In *IEEE International Real-Time Systems Symposium*, 2006.
4. T. Chantem, X. S. Hu, and M. D. Lemmon. Generalized elastic scheduling for real-time tasks. *IEEE Transactions on Computers*, 58(4):480–495, April 2009.