ILP models for the allocation of recurrent workloads upon heterogeneous multiprocessors

Sanjoy K. Baruah, Vincenzo Bonifaci, Renato Bruni, Alberto Marchetti-Spaccamela

Journal of Scheduling manuscript No.
(will be inserted by the editor)

ILP models for the allocation of recurrent workloads upon heterogeneous multiprocessors

Sanjoy K. Baruah · Vincenzo Bonifaci · Renato Bruni · Alberto Marchetti-Spaccamela

Received: date / Accepted: date

Abstract The problem of partitioning systems of independent constrained-deadline sporadic tasks upon heterogeneous multiprocessor platforms is considered. Several different integer linear program (ILP) formulations of this problem, offering different tradeoffs between effectiveness (as quantified by speedup bound) and running time efficiency, are presented. One of the formulations is leveraged to improve the best speedup guarantee known for a polynomial-time partitioning algorithm, from 12.9 to 7.83. Extensive computational results on synthetically generated instances are also provided to establish the effectiveness of the ILP formulations.

Keywords task partitioning · sporadic tasks · unrelated machines · speedup bound · ILP rounding

1 Introduction

Heterogeneous multicore CPUs – CPUs in which the processing elements differ from one another with respect to functionality or processing speed – are currently widely available and increasingly becoming the common case. The presence of such heterogeneity requires choices to be made when mapping software components onto processing elements. The need to resolve such choices adds considerable complexity to resource allocation, and inhibits the adoption of such platforms by the embedded computing industry despite significant potential benefits in terms of balancing performance and energy.

We consider here real-time systems that are modeled as collections of independent sporadic tasks (the model is described in detail in Section 2). We seek to devise algorithms for implementing such systems upon heterogeneous multiprocessor platforms under the partitioned paradigm. To our knowledge, this topic has not been studied much previously:

- On the one hand, most prior real-time scheduling research that considers heterogeneous platforms (see, e.g., [11,24,23,25,27]; [22] has a nice survey) has restricted attention to *implicit-deadline* sporadic tasks.
- On the other hand, prior research that does address the partitioned scheduling of task systems represented using models that are more general than the implicit-deadline model considers *identical* multiprocessor platforms only (see, e.g., [7,10]), or restricts the amount of heterogeneity, in the sense that the models allow only a very small number of *types* of processors (see, e.g., [4]).

In this paper, we initiate a methodical study of the problem of partitioning, upon arbitrarily heterogeneous multiprocessor platforms, task systems that are represented using the *constrained-deadline* sporadic task model[1]. We assume that once the partitioning has been performed and tasks assigned to the processors, run-

A preliminary version of this paper appeared in the Proceedings of the Euromicro Conference on Real-Time Systems, July 5-8, Toulouse (France), pp. 215–225, 2016.

S.K. Baruah
University of North Carolina at Chapel Hill.
E-mail: baruah@cs.unc.edu

V. Bonifaci
Istituto di Analisi dei Sistemi ed Informatica, CNR, Roma, Italy. E-mail: vincenzo.bonifaci@iasi.cnr.it

R. Bruni, A. Marchetti-Spaccamela
DIAG, Università di Roma "Sapienza", Roma, Italy. E-mail: {bruni, alberto}@diag.uniroma1.it

[1] Although we expect that most of our results will also extend to the *arbitrary-deadline* sporadic task model, for ease

time scheduling is done on each processor using the earliest deadline first (EDF) scheduling algorithm, which is known to be optimal for this purpose [20,12]. On the other hand, we remark that the task partitioning problem subsumes the NP-hard *unrelated machines* scheduling problem [19,26].

Our approach. We will derive various approaches to task partitioning. These algorithms share the commonality that they are all based upon formulating the task partitioning problem as an integer linear program (ILP). For implicit-deadline task systems, this is not particularly difficult to do; indeed most of the research on partitioning implicit-deadline sporadic task systems (including the works [24,23,25,27] cited above) has been based upon first formulating such an ILP, and then seeking polynomial-time algorithms for obtaining approximate solutions to these ILPs (solving an ILP is known to be NP-hard [16], and hence unlikely to be solvable exactly in polynomial time).

Despite this inherent intractability of solving ILPs, however, the optimization community has recently been devoting immense effort to devise extremely efficient implementations of ILP solvers, and highly-optimized libraries with such efficient implementations are widely available today. Modern ILP solvers, particularly when running upon powerful computing clusters, are often capable of solving ILPs with tens of thousands of variables and constraints. We therefore believe that it is reasonable to attempt to solve ILPs exactly rather than only approximately, and seek to obtain ILP formulations *that we will seek to solve exactly* to solve the partitioning problem for constrained-deadline sporadic task systems. Since the running time of ILP solvers tends to increase with the number of variables and constraints in the ILP to be solved, we seek to develop ILPs for task partitioning in which the number of variables and constraints are restricted to be low-order polynomials of the representation of the task system. While the number of constraints may not always be a good indicator of the complexity of an ILP formulation, we use it as a first approximation: indeed, the best known complexity bounds for solving ILPs do increase with the number of linear constraints [13, Theorem 5.3]. Possibly more refined metrics, such as the *constrained induced-width* [14] or the *constraint density* [2], have also been suggested in other settings, but not in the context of the problem of partitioning tasks onto heterogeneous processors – not even implicit-deadline tasks [15]; these refinements fall outside the scope of this work.

of presentation we do not explore this issue any further in this paper, but leave it for future work.

Our results. In partitioning implicit-deadline sporadic task systems, an ILP represents an exact solution to the partitioning problem — solving an ILP exactly therefore constitutes an optimal algorithm for performing such partitioning. For partitioning constrained-deadline systems, however, we do not know how to obtain such an exact ILP representation of this problem with only polynomially many constraints — this difficulty was previously identified even for partitioning upon identical multiprocessors in [5]. Instead, our goal here is to obtain polynomial-sized ILP representations of the problem of partitioning constrained-deadline sporadic task systems upon heterogeneous multiprocessor platforms with the property that exact solutions of the ILP constitute approximate solutions to the partitioning problem. Our metric of effectiveness of such an approximate solution is the *speedup factor* – an ILP formulation has speedup factor f, $f \geq 1$, if any constrained-deadline sporadic task system that can be partitioned upon a particular heterogeneous platform by a hypothetical optimal algorithm can be partitioned using this ILP formulation upon a platform in which each processor is at least f times as fast.

We have derived several different ILP representations for the problem of partitioning constrained-deadline sporadic task systems upon heterogeneous multiprocessor platforms, all of which have number of variables and constraints polynomial in the representation of the task system. All these ILP formulations have nm integer variables, where n is the number of tasks and m the number of processors, but specify different numbers of constraints and offer different speedup guarantees — they are summarized in Table 1.

Outline of the paper. The rest of the paper is structured as follows. In Section 2, we formally define our model and introduce some notation. In Section 3, we present a first ILP formulation for the partitioning problem, with a guaranteed speedup bound of 4, and show one possible generalization. In Section 4, we consider a strengthened ILP formulation that trades off the number of constraints with the speedup guarantee. Section 5 discusses a variant of the ILP of Section 3 that is used as the basis of a polynomial-time partitioning algorithm with a guaranteed speedup bound of 7.83, which improves the current bound of 12.9 [21]. In Section 6, we report the results of a large number of experimental results that test our ILP formulations on synthetically generated instances. We give our conclusions in Section 7.

Model	Number of constraints	Speedup factor	Comments
1.	$n + m + m \log_2 d_{\max}$	3	In Section 3.2
1'.	$n + m + m \log_\rho d_{\max}$	$1 + \rho$	In Section 3.3. ρ is a constant > 1. A generalization of Model 1 (which is obtained if $\rho = 2$)
2.	$n + m + n\,m\,k$	$1 + \frac{1}{k}$	In Section 4. k is any integer ≥ 1.
3.	$n + m + m \log_\rho d_{\max}$	$2 + \rho + \frac{\rho^2}{\rho - 1}$	In Section 5. A "poorer" version of Model 1' — same number of constraints, larger speedup factor. But more amenable to polynomial-time approximation – see Section 5 for details

Table 1 Summary of ILP models and results

2 System model, background, and notation

We seek to partition a sporadic task system τ comprising the n independent sporadic tasks $\tau_1, \tau_2, \ldots, \tau_n$ upon an unrelated multiprocessor platform π comprising the m processors $\pi_1, \pi_2, \ldots, \pi_m$. The i'th sporadic task τ_i is characterized by a period p_i and a relative deadline d_i, and m worst-case execution time (WCET) parameters $c_{i,1}, c_{i,2}, \ldots c_{i,m}$, with $c_{i,j}$ denoting the WCET of τ_i if it executes upon processor π_j. In this paper, we restrict attention to task systems in which $d_i \leq p_i$ for each task $\tau_i \in \tau$ — such sporadic task systems are called *constrained-deadline* sporadic task systems. During run-time, τ_i releases a sequence of jobs at time-instants that are not known beforehand, but with the constraint that successive jobs are released at least p_i time units apart. Each job of τ_i has a deadline d_i time units after its release time; the amount of execution needed by this job depends upon the identity of the processor on which it executes. More specifically, since we are studying partitioned scheduling in this paper, given task system τ and multiprocessor platform π, our objective is to obtain a partitioning of the tasks upon the processors. Let $f(i) \in \{1, 2, \ldots, m\}$ denote the index of the processor to which each τ_i is assigned under such a partitioning; each job of τ_i needs to execute for up to $c_{i,f(i)}$ time units upon processor $\pi_{f(i)}$.

Since the preemptive Earliest Deadline First scheduling algorithm (EDF) is known to be optimal for scheduling a single preemptive processor, we will use EDF as the scheduling algorithm upon each individual processor during run-time. The demand bound function (dbf) [6] of a sporadic task is widely used to quantify the computational demand of such a task, where the $\mathrm{dbf}(\tau_i, t)$ of sporadic task τ_i with period p_i, relative deadline d_i, and WCET c_i for an interval of duration t is defined as follows:

$$\mathrm{dbf}(\tau_i, t) := \left\lfloor \frac{t + p_i - d_i}{p_i} \right\rfloor c_i.$$

It is known that a collection of sporadic tasks can be scheduled to always meet all deadlines upon a preemptive uniprocessor by EDF if and only if for all $t \geq 0$, the sum of the dbf's of all the tasks in the collection for an interval of duration t does not exceed t.

Some additional notation:

- Let $N := \{1, 2, \ldots, n\}$ denote the task index set.
- Let $M := \{1, 2, \ldots, m\}$ denote the processor index set.
- Let $u_{i,j} := c_{i,j}/p_i$ denote the *utilization* of task τ_i on processor π_j.
- Let $d_{\max} := \max_{1 \leq i \leq n}\{d_i\}$ denote the largest relative deadline parameter of any task in τ.

As above, let $f : N \to M$ denote a partitioning of the task system τ upon multiprocessor platform π. We use the notation $\mathrm{dbf}_{f,j}(t)$ to denote the sum of the dbf's of all the tasks in τ that have been assigned to processor π_j under the partitioning f, for interval duration t:

$$\mathrm{dbf}_{f,j}(t) := \sum_{i \in N : f(i) = j} \left(\left\lfloor \frac{t + p_i - d_i}{p_i} \right\rfloor c_{i,j} \right).$$

3 A simple ILP model for task partitioning

Marchetti-Spaccamela et al. have previously [21] derived a polynomial-time algorithm for assigning sporadic tasks to heterogeneous processors, and shown that this algorithm has a speedup bound of $(8 + 2\sqrt{6})$ or ≈ 12.9. An intermediate step in [21] is the derivation of a 0/1 ILP representation of the partitioning problem with nm variables and $(n + m + m \log d_{\max})$ linear constraints, and a proof that this ILP representation has

a speedup factor of 6. In this section, we present two results:

1. We derive, in Theorem 1 below, an improved ILP with the same number of variables and constraints and show (Corollary 2) that it has a superior (i.e., smaller) speedup factor of 3.
2. In Theorem 2, we generalize the derivation of this improved ILP in the following manner. For any *constant* $\rho > 1$, we can derive an ILP with nm variables and $(n + m + m \log_\rho d_{\max})$ linear constraints and speedup bound 2ρ; by choosing ρ to be smaller than two, a smaller speedup bound than 3 is thus obtained at the cost of needing to solve an ILP with a larger number of constraints.

3.1 A review of some results from [21]

First a preliminary definition. Let D denote the set of values

$$\left\{0, 1, 2, 4, \ldots, 2^{\lceil \log_2 d_{\max} \rceil}\right\}.$$

We call D the *deadline checkpoint set* for task system τ.

The starting point of the reasoning in [21] is the following lemma.

Lemma 1 (from [21]) *Let $f : N \to M$ denote an assignment of the tasks in task system τ to the processors of unrelated multiprocessor platform π such that, for each $j \in M$*

$$\sum_{i:f(i)=j} u_{i,j} \leq \beta,$$

and for each $j \in M$ and $k, 1 \leq k \leq \lceil \log_2 d_{\max} \rceil$,

$$\left(\sum_{i:(f(i)=j) \wedge (2^{k-1} < d_i \leq 2^k)} c_{i,j} \right) \leq \beta \cdot 2^k.$$

Then for each $j \in M$, $\mathrm{dbf}_{f,j}(t) \leq 6\beta t$ for all $t \geq 0$.

Let us try and understand what this lemma means. The first set of inequalities requires that the cumulative utilization assigned to each processor not exceed β; the second, that the sum of the WCETs of all tasks assigned to a processor that have relative deadlines between two successive powers of two not exceed β times the larger power of two. (For example, considering $k = 6$, the second constraint mandates that the sum of the WCETs of all tasks with relative deadline in the range $(32, 64]$ not exceed 64β.) The lemma asserts that if these conditions are satisfied by an assignment, then this assignment constitutes a valid partitioning upon processors of speed $6 \times \beta$.

Now, suppose that τ is feasible upon π under partitioned scheduling, i.e., there is an assignment $f : N \to M$ of τ upon π such that all jobs of all tasks will always complete by their deadlines if tasks are assigned according to this partitioning, and each processor scheduled during run-time by EDF. For this assignment f, it is evident that the utilization constraints of Lemma 1 are satisfied for $\beta = 1$. The second set of constraints in Lemma 1 will also be satisfied for $\beta = 1$, by the following reasoning:

- Since the partitioning f is feasible, the sum of the dbf's of all the tasks assigned to the jth processor for interval duration 2^k is no more than 2^k.
- Each task with relative deadline in $(2^{k-1}, 2^k]$ must have $\mathrm{dbf}(\tau_i, 2^k) \geq c_{i,j}$.
- Hence the sum of the $c_{i,j}$'s for all tasks τ_i that have relative deadline in $(2^{k-1}, 2^k]$ and are assigned to the jth processor must be $\leq 2^k$.

Hence if τ is feasible upon π, there exists an $f : N \to M$ for which the conditions of Lemma 1 are satisfied with $\beta = 1$; therefore $\mathrm{dbf}_{f,j}(t) \leq 6t$ for all $t \geq 0$ for each j. Thus [21] derives the following speedup result:

Corollary 1 *Let $f : N \to M$ denote an assignment of tasks to processors satisfying the conditions of Lemma 1. Then f constitutes a feasible partitioning of τ upon π under a speedup factor of 6β. In particular, if the conditions of the lemma are satisfied with $\beta \leq 1/6$, then assignment f constitutes a feasible partitioning of τ upon the given platform π.*

In summary, the conditions of Lemma 1 are necessary when $\beta = 1$ and sufficient when $\beta = 1/6$, i.e., they yield a speedup factor of 6 for partitioning constrained-deadline sporadic tasks upon heterogenous multiprocessor platforms.

3.2 An ILP with speedup factor 3

We now prove an improved version of Lemma 1 that yields a superior speedup bound (of three, rather than six). The conditions specified in our theorem below differs from those in Lemma 1 only in that the summation of $c_{i,j}$'s in the second inequality is over all tasks with relative deadline $\leq 2^k$ (rather than only those with relative deadline $> 2^{k-1}$ and $\leq 2^k$).

Theorem 1 *Let $f : N \to M$ denote an assignment of the tasks in task system τ to the processors of unrelated multiprocessor platform π such that, for each $j \in M$*

$$\sum_{i:f(i)=j} u_{i,j} \leq \beta \tag{1}$$

and for each $j \in M$ *and each* $k, 1 \leq k \leq \lceil \log_2 d_{\max} \rceil$,

$$\Big(\sum_{i:(f(i)=j) \wedge (d_i \leq 2^k)} c_{i,j} \Big) \leq \beta \cdot 2^k. \quad (2)$$

Then for each j, $1 \leq j \leq m$, $\mathrm{dbf}_{f,j}(t) \leq 3\beta t$ *for all* $t \geq 0$.

Proof Consider any $t \geq 0$ and let $s := 2^k$ denote the smallest integer power of 2 that is not smaller than t (i.e. $s = 2^k \geq t > s/2$).

Consider the assignment $f : N \to M$ of the tasks to the processors as in the hypothesis of the theorem. For any $j \in M$, we have

$$\begin{aligned}
\mathrm{dbf}_{f,j}(t) &= \sum_{i:f(i)=j \wedge d_i \leq t} \left\lfloor \frac{t + p_i - d_i}{p_i} \right\rfloor c_{ij} \\
&\leq \sum_{i:f(i)=j \wedge d_i \leq t} \left(t \frac{c_{ij}}{p_i} + c_{ij} \right) \\
&\leq t \sum_{i:f(i)=j} \frac{c_{ij}}{p_j} + \sum_{i:f(i)=j \wedge d_i \leq s} c_{ij} \\
&= t \sum_{i:f(i)=j} u_{ij} + \beta \cdot 2^k \\
&\leq \beta t + \beta s \\
&\leq 3\beta t,
\end{aligned}$$

and the theorem is proved.

This implies that any f satisfying Inequalities 1 and 2 of Theorem 1 constitutes a feasible partitioning of the tasks in τ upon the set of heterogeneous processors π, when the processors receive a speedup factor of 3β.

Corollary 2 *Let* $f : N \to M$ *denote an assignment of tasks to processors satisfying Inequalities 1 and 2 of Theorem 1. Then* f *constitutes a feasible partitioning of* τ *upon* π *under a speedup factor of* 3β. *In particular, if Inequalities 1 and 2 are satisfied with* $\beta \leq 1/3$, *then assignment* f *constitutes a feasible partitioning of* τ *upon the given platform* π. □

We now apply the result of Theorem 1 above to construct a 0/1 integer linear program (ILP) for specifying a feasible partitioning of sporadic task system τ upon the platform π. That is, we will construct a 0/1 ILP, a solution to which will yield a partitioning $f : N \to M$ that satisfies Inequalities 1 and 2. This ILP is constructed as follows:

- For each $i \in N, j \in M$, we have a 0/1 integer variable (i.e., an integer variable that takes on either the value zero or the value one) $x_{i,j}$, denoting whether τ_i is to be assigned to processor π_j.
- We specify that each task gets assigned to exactly one processor; this is done by the following n constraints:

$$\forall i \in N \quad \Big(\sum_{j \in M} x_{i,j} \Big) = 1 \quad (3)$$

- We next specify that Inequality 1 of Theorem 1 should be satisfied; this is achieved by the following m constraints:

$$\forall j \in M \quad \Big(\sum_{i \in N} x_{i,j} u_{i,j} \Big) \leq \beta \quad (4)$$

- Finally, we specify Inequality 2 of Theorem 1 by the following $(\log d_{\max} \times m)$ constraints:

$$\forall k \in D, \forall j \in M \Big(\sum_{(i \in N) \wedge (d_i \leq 2^k)} c_{i,j} x_{i,j} \Big) \leq \beta \cdot 2^k \quad (5)$$

By Theorem 1, any assignment of integer values to the $\{x_{i,j}\}$ variables satisfying the Constraints 3–5 above bears witness to the schedulability of τ, with a speedup of 3β, upon the platform π. Moreover, for a τ that is feasible upon π the model always admits a solution with $\beta = 1$, since all inequalities are clearly valid; thus, this ILP guarantees a speedup factor of at most 3. When the ILP model is feasible with $\beta \leq 1/3$, Theorem 1 guarantees schedulability on the original platform; hence, a reasonable objective function for the ILP with Constraints (3)–(5) would be to **minimize** β.

Throughout the rest of this paper, we will still refer to the problem of the minimization of β subject to (3)–(5) as ILP, even if β is not strictly constrained to be integer. However, we adopt this terminology for extension, and also because β could equivalently be fixed, for example in a binary search fashion.

The ILP model consisting of Contraints (3)–(5) will be referred to as *Model 1* in the remainder of the paper.

3.3 A generalization

Above, we derived an ILP model for the problem of partitioned scheduling of constrained-deadline sporadic task systems upon unrelated multiprocessors, such that any solution to the ILP immediately yields a partitioning algorithm at a speedup bound of 4. We also saw that this ILP has $(n + m + m \times \lceil \log_2 d_{\max} \rceil)$ constraints; we now briefly describe how to reduce the speedup bound by increasing the number of constraints.

Recall that we had defined the *deadline checkpoint set* D as powers of two: $D = \{0, 2^0, 2, 2^2, \ldots, 2^{\lceil \log_2 d_{\max} \rceil}\}$. For any given constant $\rho > 1$, we could instead have chosen to define it as

$$D_\rho = \{0, 1, \rho, \rho^2, \rho^3, \ldots, \rho^{\lceil \log_\rho d_{\max} \rceil}\},$$

The following generalization of Theorem 1 is easily proved via a proof analogous to the proof of Theorem 1:

Theorem 2 *Let $f : N \rightarrow M$ denote an assignment of the tasks in task system τ to the processors of unrelated multiprocessor platform π such that, for each $j \in M$*

$$\sum_{i:f(i)=j} u_{i,j} \leq \beta,$$

and for each $j \in M$ and $k, 1 \leq k \leq \lceil \log_\rho d_{\max} \rceil$,

$$\Big(\sum_{i \in N:(f(i)=j) \wedge (d_i \leq \rho^k)} c_{i,j} \Big) \leq \beta \cdot \rho^k$$

Then for each $j \in M$, $\mathrm{dbf}_{f,j}(t) \leq (1+\rho)\beta t$ for all $t \geq 0$. □

It follows, from arguments virtually identical to those of Corollary 2, that the speedup bound for the ILP constructed based on Theorem 2 above is $(1+\rho)$; hence by choosing ρ to be smaller than 2 a speedup bound smaller than 3 is obtained. The tradeoff is that the number of constraints increases to $(n+m+m \times \lceil \log_\rho d_{\max} \rceil)$; this is $> \log_2 d_{\max}$ for $\rho < 2$, becoming larger as $\rho \rightarrow 1$.

Theorem 2 thus suggests one approach for obtaining speedup bounds arbitrarily close to 2, by simply selecting a denser deadline checkpoint set. In Section 4 below, we explore another approach, that allows for speedup bounds arbitrarily close to one (once again at the cost of having additional constraints).

4 A strengthened ILP formulation

We now explore a different idea that also trades off an increase in the number of constraints in the ILP for a superior speedup bound. Specifically, for any positive integer constant k we will derive an ILP model with $(n + m + m\,n\,k)$ constraints, finding a feasible solution to which corresponds a partitioning at a speedup bound of $\left(1 + \frac{1}{k}\right)$.

Approximation schemes have been defined for computing the value of dbf to any desired degree of accuracy (see, e.g. [3]). Equation 6 below gives such an approximation scheme; for any fixed positive integer value of k, $\mathrm{dbf}^{(k)}(\tau_i, t)$ defines an approximation of $\mathrm{dbf}(\tau_i, t)$ that is exact for the first k "steps" of $\mathrm{dbf}(\tau_i, t)$, and an upper bound for larger values of t:

$$\mathrm{dbf}^{(k)}(\tau_i, t) = \begin{cases} c_{i,j} \times \lfloor \frac{t+p_i-d_i}{p_i} \rfloor & \text{if } t \leq (k-1)p_i + d_i \\ c_i + (t - d_i)u_i & \text{otherwise} \end{cases} \tag{6}$$

The following lemma (see, e.g., [5]) provides a quantitative bound on the degree by which $\mathrm{dbf}^{(k)}$ may deviate from dbf:

Lemma 2 *For all $t \geq 0$*

$$\mathrm{dbf}(\tau_i, t) \leq \mathrm{dbf}^{(k)}(\tau_i, t) < \left(1 + \frac{1}{k}\right) \mathrm{dbf}(\tau_i, t).$$

That is, $\mathrm{dbf}^{(k)}(\tau_i, t)$ provides an upper bound on $\mathrm{dbf}^{(k)}(\tau_i, t)$ that is no more than a fraction $1/k$ greater than the actual value of $\mathrm{dbf}^{(k)}(\tau_i, t)$.

As previously stated, it is known that a collection of sporadic tasks can be scheduled to always meet all deadlines upon a preemptive uniprocessor by EDF if and only if for all $t \geq 0$, the sum of the dbf's of all the tasks in the collection over an interval of duration t does not exceed t. For schedulability, it is clearly necessary that the utilizations of all the tasks in the collection sum to no more than 1. Since $\mathrm{dbf}^{(k)}$is an upper bound on dbf, it follows that a *sufficient* uniprocessor EDF-schedulability test for a collection of tasks is that the sum of the $\mathrm{dbf}^{(k)}$ functions of all the tasks in the collection over an interval of duration t not exceed t. Albers and Slomka showed [3, Lemma 4] that it suffices to validate this fact only for those values of t at which one or more of the $\mathrm{dbf}^{(k)}$ functions has a step discontinuity. More precisely, let

$$S_{i,k} = \{t : t = d_i + hp_i, h = 0, 1, \ldots, k\}$$

and $S_k = \bigcup_{\text{all } i} S_{i,k}$

It suffices to test that the sum of the $\mathrm{dbf}^{(k)}$ functions of all the tasks in the collection over an interval of duration t not exceed t, only for values of $t \in S_k$.

We can use this result to define a revised ILP formulation for modeling the partitioned scheduling of sporadic task systems upon heterogeneous multiprocessors. The first part of this revised ILP is identical to the one constructed in Section 3:

- As in Section 3, we will have a zero-one integer variable $x_{i,j}$, denoting whether τ_i is to be assigned to processor π_j, for each $i \in N, j \in M$.
- Again as in Section 3, the following n constraints specify that each task gets assigned to exactly one processor:

$$\forall i \in N \quad \Big(\sum_{j \in M} x_{i,j} \Big) = 1 \tag{7}$$

- The following m constraints bound the total utilization on each processor:

$$\forall j \in M \quad \Big(\sum_{i \in N} x_{i,j} u_{i,j} \Big) \leq \beta \tag{8}$$

The final set of constraints replace the Inequalities 5 of the ILP in Section 3 with constraints based upon the $\mathrm{dbf}^{(k)}$ approximation, that express the requirement

that the sum of the $\mathrm{dbf}^{(k)}$ functions of all tasks assigned each processor over an interval duration not exceed the duration. As we had stated above, this condition only needs to be validated for interval durations in S_k; this motivates the following set of constraints:

$$\forall t \in S_k,\ \forall j \in M \left(\sum_{i \in N} \left(x_{i,j} \times \mathrm{dbf}_j^{(k)}(\tau_i, t) \right) \right) \leq \beta \cdot t \quad (9)$$

(where for each $j \in M$, $\mathrm{dbf}_j^{(k)}(\tau_i, t)$ denotes the function $\mathrm{dbf}^{(k)}(\tau_i, t)$ when the WCET of τ_i is set equal to $c_{i,j}$.) Since each τ_i contributes at most k distinct points to S_k, it follows that $|S_k| \leq nk$; hence there are at most mnk such constraints.

Note that the Inequalities 9 are constructed for specified values of t; i.e., for each $t \in S_k$. For each such specified t, it is straightforward to observe that the inequality is indeed a linear one, since inspection of Expression 6 reveals that for a given value of t for each τ_i the expression $\mathrm{dbf}_j^{(k)}(\tau_i, t)$ is a *constant*.

Theorem 3 *A feasible solution to the ILP on the 0/1 variables $\{x_{i,j}\}, i \in N, j \in M$, with Constraints 7–9 (defined above) yields a feasible partitioning of the tasks in τ to a platform in which each processor in π is speeded up by a multiplicative factor of $(1 + 1/k)\beta$. In particular, if the inequalities are satisfied with $\beta \leq (1 + 1/k)^{-1}$, then a feasible solution to the ILP yields a feasible partitioning on the given platform.*

Proof It is evident from the result of Albers and Slomka that satisfying Constraints 7–9 is sufficient for feasibility upon a speed-$\beta(1 + 1/k)$ platform. Additionally we conclude from the lower bound in Lemma 2 that failure to satisfy these conditions implies infeasibility upon a speed-β platform.

The ILP model consisting of Constraints (7)–(9) will be referred to as *Model 2* in the remainder of the paper. The quality of the solution that is obtained by solving Model 2 depends on the value of k: the larger this value, the better is the quality (i.e., the lower the speedup factor) of the obtained solution. However we observe that the number of constraints increases with k. It follows that large values of k lead to an ILP that is not solvable with state of the art packages.

5 An ILP that is amenable to polynomial time approximation

In the previous sections, we discussed assignment algorithms based on solving reasonably-sized ILPs. However, in some scenarios the solution of an ILP may be a computational bottleneck – solving an ILP is, after all, a prototypical NP-hard problem. Therefore, the design of efficient (polynomial time) assignment and schedulability algorithms retains interest.

A standard technique that has been developed for efficiently obtaining an approximate solution to an ILP is to first consider the linear program (LP) obtained by "relaxing" (i.e., ignoring) the integrality requirement, solve this LP (this can be done in polynomial time), and then "rounding" the solution so obtained to obtained an integral solution as desired. The main challenge in designing the rounding procedure is to ensure that such rounding does not degrade the feasibility or the quality of the solution (i.e., the value of the objective function that was optimized) too much.

Recently, a new approach to rounding, known as *iterative rounding* [18], has been shown to provide improved rounding guarantees. Analogously to prior rounding approaches, the first step requires that an LP relaxation be solved and a non-integer solution (say, X) be obtained. However, instead of rounding all non integral values of X at the same time, only one variable is rounded; assume, for example, that the value of variable x_1 is set to $\hat{x}_1$. Then the method iterates and solves a new LP-relaxation that is obtained from the original LP by fixing the value of x_1 to $\hat{x}_1$. In this way, a new solution X' is obtained; as in the previous case, the method now rounds one variable of X'; the method is iterated until all variables satisfy the given integrality constraints.

In this section, we seek to construct an ILP formulation of the problem of partitioning sporadic tasks upon heterogeneous multiprocessors that is more amenable to iterative rounding than the ILP formulations we have seen above. It will turn out (Theorem 4 below) that this ILP has the same number of variables and constraints, but a poorer (larger) speedup factor than the one described in Section 3.3 (Theorem 2). Hence from the perspective of just developing an ILP, this is not a particularly useful result. However, we will see that this ILP can in fact be rounded iteratively in a manner that we were unable to pull off with the earlier ILP formulations, resulting in a polynomial-time algorithm for partitioning sporadic tasks upon heterogeneous multiprocessors that has speedup bound of ≈ 7.83, thereby improving the ≈ 12.9 speedup bound of Marchetti-Spaccamela et al. [21].

5.1 LP-rounding based approach

As before, we use variables x_{ij} for each pair $(i, j) \in N \times M$, modeling the assignment of τ_i to π_j. Apart from the usual assignment constraints, the first constraints

we consider are the utilization bounds on the tasks assigned to the same processor. That is, we require that

$$\sum_{i \in N} u_{ij} x_{ij} \leq 1 \qquad \forall j \in M. \tag{10}$$

Now, let ρ denote any constant, $\rho > 1$. We define the function $r(x) := \rho^{\lceil \log_\rho x \rceil}$, and the set $D_\rho := \{\rho^0, \rho^1, \ldots, r(d_{\max})\}$. We want to express the requirement that for all tasks assigned to the same processor with deadline at most ρ^k, the sum of their WCETs is at most ρ^k. Note that this is the set of tasks with $r(d_i) \leq \rho^k$. For technical reasons that will become apparent later (in Lemma 3), we adopt the slightly weaker constraint

$$\sum_{i \in N:\, r(d_i) \leq d} c_{ij} \left(1 - \frac{d_i}{p_i}\right) x_{ij} \leq d \quad \forall d \in D_\rho, \forall j \in M. \tag{11}$$

We call these constraints (11) the *relaxed* dbf *constraints*. It is clear that these constraints have to be fulfilled by any feasible task assignment. (In particular, if $d_i \leq \rho^k$ and $x_{ij} = 1$, then $\mathrm{dbf}_{f,j}(\rho^k) \geq c_{ij} > c_{ij}(1 - d_i/p_i)$, where f is the assignment represented by $\mathbf{x}$). We therefore arrive at the following ILP, denoted poly-ILP.

$$\sum_{j \in M} x_{ij} = 1 \qquad \forall i \in N \tag{12a}$$

$$\sum_{i \in N} u_{ij} x_{ij} \leq 1 \qquad \forall j \in M \tag{12b}$$

$$\sum_{i \in N:\, r(d_i) \leq d} c_{ij} \left(1 - \frac{d_i}{p_i}\right) x_{ij} \leq d \qquad \forall d \in D_\rho, \forall j \in M \tag{12c}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in N, \forall j \in M \text{ s.t. } c_{ij} \leq d_i. \tag{12d}$$

If poly-ILP is infeasible, then there can be no feasible task assignment. Now assume that it is feasible and consider its relaxation, which is obtained by replacing each constraint (12d) by

$$x_{ij} \geq 0 \quad \forall i \in N, \forall j \in M \text{ s.t. } c_{ij} \leq d_i. \tag{13}$$

Since it is an LP and not an ILP, the relaxation can be solved in polynomial time. Let $\mathbf{x}^*$ denote its solution. For each $j \in M$ and deadline $d \in D_\rho$, we compute the value

$$b_{j,d} := \sum_{i \in N:\, r(d_i) = d} c_{ij}(1 - d_i/p_i) x^*_{ij}.$$

Note that, by (12c),

$$\sum_{d' \leq d} b_{j,d'} \leq d \qquad \forall d \in D_\rho, \forall j \in M. \tag{14}$$

Based on these computed values, we define a variation of poly-ILP, denoted by sparse-ILP in the sequel. We obtain the latter by replacing the constraints (12c) with the following set of constraints:

$$\sum_{i \in N:\, r(d_i) = d} c_{ij} \left(1 - \frac{d_i}{p_i}\right) x_{ij} \leq b_{j,d} \quad \forall d \in D_\rho, \forall j \in M.$$

By dividing both sides of the inequality by d, these constraints can also be written as

$$\sum_{i \in N:\, r(d_i) = d} \bar{c}_{ijd} \left(1 - \frac{d_i}{p_i}\right) x_{ij} \leq \bar{b}_{j,d} \quad \forall d \in D_\rho, \forall j \in M. \tag{15}$$

where $\bar{b}_{j,d} = b_{j,d}/d \leq 1$ (since $\mathbf{x}^*$ is feasible for the relaxation of poly-ILP) and $\bar{c}_{ijd} = c_{ij}/d \leq 1$ (since if $r(d_i) = d$ then $c_{ij} \leq d_i \leq d$). Let $\mathcal{A}$ be the set of vectors $\mathbf{x}$ satisfying (12a). We can now express sparse-ILP in matrix notation as

$$\{\mathbf{x} \in \mathcal{A} \cap \{0,1\}^{N \times M} \,:\, \mathbf{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}\},$$

where $\mathbf{A}$ and $\mathbf{b}$ are, respectively, the matrix of coefficients and the vector of right hand sides of the constraints (12b) and (15). Note that all entries of $\mathbf{A}$ and $\mathbf{b}$ take values between 0 and 1.

By construction, if $\mathbf{x}^*$ is a feasible solution for the LP relaxation of poly-ILP it is also feasible for the LP relaxation of sparse-ILP, and if the LP relaxation of sparse-ILP is infeasible, then no feasible task assignment exists. Our goal will be to round $\mathbf{x}^*$ to an *integral* vector which *approximately* satisfies the constraints of sparse-ILP.

A reason for preferring sparse-ILP to poly-ILP is that the former is an ILP formulation in which the constraint matrix is sparse: each variable appears in only a small number of constraints. This *sparsity* gives the potential to derive efficient rounding schemes which result in integral solutions, violating the relaxed dbf-constraints only by constant factors. We present such a rounding scheme below; to this end, the following theorem shows that—even when violated up to constant factors–the relaxed dbf constraints (15), together with the utilization constraints (12b), are approximately sufficient.

Theorem 4 *Let $\beta \geq 1$ and let $f : N \to M$ be an assignment encoded by a vector $\hat{\mathbf{x}} \in \mathcal{A} \cap \{0,1\}^{N \times M}$ such that, for each $j \in M$ and $d \in D_\rho$,*

$$\sum_{i \in N} u_{ij} \hat{x}_{ij} \leq \beta \tag{16}$$

and

$$\sum_{i\in N\,:\,r(d_i)=d} c_{ij}\left(1-\frac{d_i}{p_i}\right)\hat{x}_{ij} \le b_{j,d}+(\beta-1)\cdot d. \qquad (17)$$

Then $\mathrm{dbf}_{f,j}(s) \le (\beta+\rho+(\beta-1)\rho^2/(\rho-1))s$ *for all* $s \ge 0$. *In particular, if* $\beta \le 2$, f *is a feasible assignment under a speedup factor of* $(2+\rho+\rho^2/(\rho-1))$.

Proof For any $s \ge 0$ and $j \in M$, we bound $\mathrm{dbf}_{f,j}(s)$ as follows:

$$\begin{aligned}
\mathrm{dbf}_{f,j}(s) &= \sum_{i\in N:\, d_i\le s} \lfloor\frac{s+p_i-d_i}{p_i}\rfloor c_{ij}\hat{x}_{ij}\\
&\le \sum_{i\in N:\, d_i\le s}\left(s\frac{c_{ij}}{p_i}+c_{ij}\left(1-\frac{d_i}{p_i}\right)\right)\hat{x}_{ij}\\
&\le \sum_{i\in N:\, r(d_i)\le r(s)}\left(s\frac{c_{ij}}{p_j}+c_{ij}\left(1-\frac{d_i}{p_i}\right)\right)\hat{x}_{ij}\\
&\le s\sum_{i\in N}\frac{c_{ij}}{p_i}\hat{x}_{ij}+\sum_{i\in N:\, r(d_i)\le r(s)} c_{ij}\left(1-\frac{d_i}{p_i}\right)\hat{x}_{ij}\\
&= s\sum_{i\in N} u_{ij}\hat{x}_{ij}+\sum_{k=0}^{\log_\rho(r(s))}\sum_{i\in N:\, r(d_i)=\rho^k} c_{ij}(1-\frac{d_i}{p_i})\,\hat{x}_{ij}\\
&\overset{(16)}{\le} \beta s+\sum_{k=0}^{\log_\rho(r(s))}\sum_{i\in N:\, r(d_i)=\rho^k} c_{ij}(1-\frac{d_i}{p_i})\,\hat{x}_{ij}\\
&\overset{(17)}{\le} \beta s+\sum_{k=0}^{\log_\rho(r(s))}\left(b_{j,\rho^k}+(\beta-1)\rho^k\right)\\
&\overset{(14)}{\le} \beta s+\rho^{\log_\rho(r(s))}+\sum_{k=0}^{\log_\rho(r(s))}(\beta-1)\rho^k\\
&= \beta s+r(s)+(\beta-1)\sum_{k=0}^{\log_\rho(r(s))}\rho^k\\
&= \beta s+r(s)+(\beta-1)\cdot\frac{\rho^{\log_\rho(r(s))+1}-1}{\rho-1}\\
&= \beta s+r(s)+(\beta-1)\frac{\rho\cdot r(s)-1}{\rho-1}\\
&\le (\beta+\rho+(\beta-1)\frac{\rho^2}{\rho-1})s.
\end{aligned}$$

The last inequality follows from $r(s) \le \rho\cdot s$.

To construct $\hat{\mathbf{x}}$, we adopt an iterative rounding procedure that is similar to the procedure presented in [17, 21]. The idea of the iterative rounding procedure is the following. In each iteration k, we first compute an extreme point solution $\mathbf{x}^k$ of a linear program LP^k, where LP^0 is the relaxation of sparse-ILP, and each LP^k is obtained by fixing the value for some variables or removing some constraints of LP^{k-1}.

Given a feasible fractional solution $\mathbf{x}^k$, to define LP^{k+1} we first fix all variables which are integral in $\mathbf{x}^k$, i.e., those variables are not allowed to be changed anymore in the remainder of the procedure. Let v be the number of variables in LP^k and let w_a, w_b and w_b be the number of constraints of types (12a), (12b) and (15), respectively. Let $w = w_a + w_b + w_c$. To obtain LP^{k+1} we either delete one or more variables, in case $v > w$, or delete a constraint while ensuring that in the final solution that constraint will not be violated too much. Along the way we ensure that the constraints of type (12a) are always satisfied exactly, so that $\mathbf{x}^k \in \mathcal{A}$ at all times.

Note that if there is some variable x_{ij} that is fixed at value 1 and removed from the program, then for all $j' \in M \setminus \{j\}$, $x_{ij'}$ will be set to 0 and also be removed from the program. The constraint of type (12a) corresponding to this i is then superfluous and will also be removed.

To derive the bounds (16)–(17), we need to study the coefficient matrix $\mathbf{A}$ in more detail. Let γ be the maximum, over all x_{ij}, of the sum of the values of the coefficients of variable x_{ij} in constraints (12b) and (15). We first derive a bound on γ.

Lemma 3 *For any task set* τ, $\gamma \le 1$.

Proof Observe that γ is just the maximum value of $u_{ij}+\bar{c}_{ijd}(1-d_i/p_i)$ across all variables x_{ij} in the program. Recall that for all such pairs (i,j), $c_{ij} \le d_i$, i.e., $\bar{c}_{ijd} \le 1$, otherwise the variable x_{ij} is forced to 0 and removed from the LP. We can now bound

$$u_{ij}+\bar{c}_{ijd}\left(1-\frac{d_i}{p_i}\right) \le \frac{c_{ij}}{p_i}+1-\frac{d_i}{p_i} \le \frac{c_{ij}}{p_i}+1-\frac{c_{ij}}{p_i} = 1.$$

The following technical lemma is instrumental to our rounding procedure. It is a specialization to our setting of a more general rounding result for assignment LPs [9].

Lemma 4 *Let* LP^k *be the linear program that is solved in iteration* k *of the rounding procedure, with* s *variables and* r *constraints. Let* $\mathbf{x}^k$ *be an extreme point solution to this LP. Then either,*
(i) $\mathbf{x}^k$ *has at least one integer component; or*
(ii) there is $j \in M$ *and a corresponding constraint of type* (12b) *such that* $\sum_{i\in N} u_{ij}z_{ij} - \sum_{i\in N} u_{ij}x^k_{ij} \le \gamma$ *for* any *integer solution* $\mathbf{z}$*; or*
(iii) there are $j \in M$, $d \in D_\rho$ *and a corresponding constraint of type* (15) *such that*

$$\sum_{i\in N:\, r(d_i)=d}\bar{c}_{ijd}\left(1-\frac{d_i}{p_i}\right)(z_{ij}-x^k_{ij}) \le \gamma$$

for any *integer solution* $\mathbf{z}$.

Proof Let $\mathbf{A}$ be the coefficient matrix of LP^k. If $v > w$, the null space of $\mathbf{A}$ is nontrivial, so let $\mathbf{x}_0$ be a nonzero vector in the null space. Since $\mathbf{x}^k$ is an extreme-point solution to LP^k, it cannot be expressed as the convex combination of two (or more) solutions to LP^k. If $\mathbf{x}^k$ does not have any integral entry, then we can find a value $\delta > 0$ such that $\mathbf{x}^k + \delta\mathbf{x}_0$ and $\mathbf{x}^k - \delta\mathbf{x}_0$ are both solutions to LP^k (since $\mathbf{A}(\mathbf{x} \pm \delta\mathbf{x}_0) = \mathbf{A}\mathbf{x}$) and, in particular, $\mathbf{x}^k$ is a convex combination of these two solutions. Therefore $\mathbf{x}^k$ must have at least one integral entry.

If $v \leq w$, we show that there always exists a constraint of type (12b) or type (15) such that $\max_{\mathbf{z}\in S}\{(\mathbf{A}\mathbf{z})_l - (\mathbf{A}\mathbf{x})_l\} \leq \gamma$, where γ is the maximum sum of coefficients in a column of constraints (12b) and (15) and where S is the integer solution space for all remaining variables, i.e., $S = \{0,1\}^v$.

We show the statement by contradiction. Assume that the statement is not true, that is, for each constraint l of type (12b) or (15) it holds that there exists a vector $\mathbf{z} \in S$ such that

$$(\mathbf{A}\mathbf{z})_l - (\mathbf{A}\mathbf{x})_l > \gamma. \tag{18}$$

Note that all variables still present in the linear program correspond to a processor $j \in M$ and a task $i \in N$ that is not yet assigned fully to one processor, but fractionally to multiple processors. Hence, the constraint of type (12a) corresponding to each τ_i is still present in the linear program. It follows that

$$\sum_{j\in M} \sum_{i\in N:\, x_{ij}\in(0,1)} x_{ij} = w_a, \tag{19}$$

where w_a is the number of constraints of type (12a) remaining in LP^k. Define L as the set of constraints of types (12b) and (15) present in the current linear program, and let w_b and w_c be their number, respectively (so $w_a + w_b + w_c = w$). For any $q = (i,j)$, let L^q denote the set of these constraints containing variable x_q; by definition of γ, we have

$$\max_q \sum_{l\in L^q} a_{lq} \leq \gamma. \tag{20}$$

Then,

$$\begin{aligned}
\gamma(w - w_a) &= \gamma(w_b + w_c) \\
&\overset{(18)}{<} \sum_{l\in L} \max_{\mathbf{z}\in S}((\mathbf{A}\mathbf{z})_l - (\mathbf{A}\mathbf{x})_l) \\
&\overset{\text{as all } a_{lq}\geq 0}{=} \sum_{l\in L}((\mathbf{A}\mathbf{1})_l - (\mathbf{A}\mathbf{x})_l) \\
&= \sum_{l\in L}\sum_q a_{lq}(1 - x_q) \\
&= \sum_q \sum_{l\in L^q} a_{lq}(1 - x_q) \\
&\leq \sum_q \gamma(1 - x_q) \\
&= \gamma v - \sum_q \gamma x_q \\
&= \gamma(v - r_a).
\end{aligned} \tag{21}$$

The second inequality follows from (20).

The chain of inequalities implies that $\gamma(w - w_a) < \gamma(w - w_a) \Rightarrow r < s$ which is a contradiction to being in the case that $v \leq w$. Hence we conclude that if $v \leq w$, there must be a constraint l of type (12b) or (15) for which $\max_{\mathbf{z}\in S}\{(\mathbf{A}\mathbf{z})_l - (\mathbf{A}\mathbf{x})_l\} \leq \gamma$.

Lemma 4 is used to guide the rounding process. If Case (i) applies, the variables that have an integer value are fixed at that value and removed from the LP. If a variable x_{ij} is fixed at value 1, then for all $j' \in M\backslash\{j\}$, the variables $x_{ij'}$ are fixed at value 0 and the constraint of type (12a) corresponding to i is removed. If we are in Case (ii) or (iii), the constraint for which the claim holds can be found in polynomial time by checking, for each constraint $l \in L$ of type (12b) or (15), whether $\sum_q a_{lq}(1 - x_q) \leq \gamma$. This is sufficient since all $a_{lq} \geq 0$ and the maximum value any variable x_q can take is 1. If such a constraint is of type (12b) (Case (ii)) or (15) (Case (iii)), the final task assignment will satisfy (16) or (17) for that constraint, respectively, *even if the constraint is dropped*; thus, we drop the constraint, obtaining the next (smaller) LP.

After either all constraints have been removed or the values of all variables have been fixed at an integer value, we obtain an integral vector $\hat{\mathbf{x}}$ which satisfies $\sum_{i\in N} u_{ij}\hat{x}_{ij} \leq 1 + \gamma$ for each $j \in M$ and

$$\sum_{i\in N:\, r(d_i)=d} \bar{c}_{ijd}\left(1 - \frac{d_i}{p_i}\right) x_{ij} \leq \bar{b}_{j,d} + \gamma$$

for all $j \in M$ and all deadlines $d \in D_\rho$. Hence, the vector $\hat{\mathbf{x}}$ satisfies constraints (16), (17) with $\beta := 1 + \gamma$. We are now in the position to invoke Theorem 4 to obtain our final guarantee.

Theorem 5 *There is a polynomial-time partitioning algorithm with a speedup bound of* $(5 + 2\sqrt{2}) \approx 7.83$ *for the problem of assigning constrained-deadline tasks to heterogeneous processors.*

Proof All steps required to construct $\hat{\mathbf{x}}$ can be carried out in polynomial time. The assignment induced by $\hat{\mathbf{x}}$ satisfies (16)–(17) with $\beta = 1 + \gamma \leq 2$ (Lemma 3).

Thus, by Theorem 4 with $\rho = 1+1/\sqrt{2}$, the assignment induced by $\hat{\mathbf{x}}$ is feasible with speedup

$$2 + \rho + \frac{\rho^2}{\rho - 1} = 5 + 2\sqrt{2} \approx 7.83.$$

□

5.2 Practical variant of the LP-rounding model

The above LP-rounding approach was designed with the aim of minimizing the worst-case speedup bound in Theorem 5. However, we observed that certain steps required for theoretical soundness –namely, sparsification of the constraints– can be avoided in practice. Thus, in this subsection we briefly discuss the practical variant of the LP-rounding model that is the one we adopted in the schedulability experiments, and that we will refer to as *Model 3*.

The starting point is the following minimization version of equations (12a)-(12d):

$$\min \beta \tag{22a}$$

$$\sum_{j \in M} x_{ij} = 1 \qquad \forall i \in N \tag{22b}$$

$$\sum_{i \in N} u_{ij} x_{ij} \leq \beta \qquad \forall j \in M \tag{22c}$$

$$\sum_{i \in N:\, r(d_i) \leq d} \frac{c_{ij}}{d} \left(1 - \frac{d_i}{p_i}\right) x_{ij} \leq \beta \qquad \forall d \in D_\rho,\, \forall j \in M \tag{22d}$$

$$x_{ij} \in \{0,1\} \qquad \forall i \in N, \forall j \in M \text{ s.t. } c_{ij} \leq d_i, \tag{22e}$$

where β is the speedup parameter to be minimized. We apply to this model the iterative rounding procedure as described in the previous subsection, except that when we need to drop a constraint, we drop the constraint l for which the potential violation (i.e., $\sum_q a_{lq}(1 - x_q)$ where A is the coefficient matrix of Constraints (22c)–(22d)) is the smallest. We also keep track of the largest potential violation value during the whole process (we call it γ, in analogy with the previous section). To determine whether the instance is schedulable, at the end of the process it is sufficient to compare the final value of $\beta+\gamma$ against $(1+\rho)^{-1}$, due to the following Theorem.

Theorem 6 *Let $\beta, \gamma > 0$ and let $f : N \to M$ be an assignment encoded by a vector $\hat{\mathbf{x}} \in \mathcal{A} \cap \{0,1\}^{N \times M}$ such that, for each $j \in M$ and $d \in D_\rho$,*

$$\sum_{i \in N} u_{ij} \hat{x}_{ij} \leq \beta + \gamma \tag{23}$$

and

$$\sum_{i \in N\,:\, r(d_i) \leq d} c_{ij} \left(1 - \frac{d_i}{p_i}\right) \hat{x}_{ij} \leq (\beta + \gamma) d. \tag{24}$$

Then $\mathrm{dbf}_{f,j}(s) \leq (1+\rho)(\beta+\gamma)s$ for all $s \geq 0$. In particular, if $\beta + \gamma \leq (1+\rho)^{-1}$, then f is a feasible assignment on unit speed processors.

Proof The proof is similar to that of Theorem 4. For any $s \geq 0$ and $j \in M$, we bound $\mathrm{dbf}_{f,j}(s)$ as follows:

$$\begin{aligned}
\mathrm{dbf}_{f,j}(s) &= \sum_{i \in N:\, d_i \leq s} \lfloor \frac{s + p_i - d_i}{p_i} \rfloor c_{ij} \hat{x}_{ij} \\
&\leq \sum_{i \in N:\, d_i \leq s} \left(s \frac{c_{ij}}{p_i} + c_{ij} \left(1 - \frac{d_i}{p_i}\right) \right) \hat{x}_{ij} \\
&\leq \sum_{i \in N:\, r(d_i) \leq r(s)} \left(s \frac{c_{ij}}{p_j} + c_{ij} \left(1 - \frac{d_i}{p_i}\right) \right) \hat{x}_{ij} \\
&\leq s \sum_{i \in N} \frac{c_{ij}}{p_i} \hat{x}_{ij} + \sum_{i \in N:\, r(d_i) \leq r(s)} c_{ij} \left(1 - \frac{d_i}{p_i}\right) \hat{x}_{ij} \\
&= s \sum_{i \in N} u_{ij} \hat{x}_{ij} + \sum_{i \in N:\, r(d_i) \leq r(s)} c_{ij} \left(1 - \frac{d_i}{p_i}\right) \hat{x}_{ij} \\
&\leq s(\beta + \gamma) + (\beta + \gamma) r(s) \\
&\leq s(\beta + \gamma) + \rho s (\beta + \gamma) \\
&= (1 + \rho)(\beta + \gamma) s.
\end{aligned}$$

The last inequality follows from $r(s) \leq \rho \cdot s$.

6 Schedulability experiments

In the sections above, we saw how the problem of partitioned scheduling of sporadic task systems upon unrelated multiprocessors could be modeled by ILPs. Our motivation for doing so is that the optimization community has devoted immense effort to coming up with extremely efficient (although still exponential-time, since solving ILPs is NP-hard) algorithms for solving ILPs, and highly-optimized libraries implementing these efficient algorithms are widely available. This is particularly true for ILPs like the ones we have constructed above, in which each variable is further constrained to take in only the values zero or one. In this section, we validate the performance of our ILP-based schedulability tests against synthetic workloads in terms of the percentage of schedulable task sets.

Our results show that Model 2 (the ILP model discussed in Section 4) provides solutions having better quality, while Model 1 (the ILP model discussed in Section 3) generates solutions of lower quality but in less time. Model 3, the polynomial time LP-rounding approach discussed in Section 5 has a solution quality comparable to that of the Model 1, but runs even faster.

6.1 Generation of the task sets and solutions

We developed a parametric framework with several parameters (m, κ, p, $\bar{U}$, α – they are detailed below) to randomly generate our workloads; this framework is general enough to support our entire range of experiments.

We consider m-processor platforms and $n = \kappa m$ tasks, with $\kappa \geq 1$ an integer-valued parameter. We randomly generate an affinity mask $R_{i,j}$ for each $i \in N$ and $j \in M$: $R_{i,j} \leftarrow 1$ with probability p and 0 with probability $(1-p)$. These affinity masks help define the $C_{i,j}$ values: $C_{i,j}$ has a value $< \infty$ if and only if $R_{i,j} = 1$. If the generated mask does not allow a particular task to be processed on any processor, we then allow that task to be processed upon a randomly chosen processor.

We then generate utilization values for every allowed pair (i,j) for which $R_{i,j} = 1$. Tasks are grouped into m groups of size κ each: tasks $\tau_{1+(k-1)\kappa}$ to $\tau_{k\kappa}$ form the kth group. For each group we use the UUniSort algorithm [8] to generate randomly distributed utilizations with total value $\bar{U}$ for the allowed task-processor pairs in the group. Note that since there are m groups, each of total utilization $\bar{U}$, the value $\bar{U}$ represents the average load that a processor can expect if tasks are randomly assigned. Note that $\bar{U} \leq 1$ is *not* a necessary condition for schedulability (indeed some of our ILP formulations are able to schedule task sets with $\bar{U} > 1$).

We generate the periods by setting $p_i = 2^{\Delta_i}$, with each Δ_i uniformly distributed in the range 3...10. The worst-case execution times are computed directly from the periods and utilizations. Finally, the relative deadline of each task is sampled uniformly in the range $[(1-\alpha)\cdot(\max_j c_{ij}) + \alpha p_i, p_i]$, where $\alpha \in [0,1]$.

In the experiments we consider the three models: Model 1 (that is, the ILP model discussed in Section 3.2), Model 2 (that is, the ILP model discussed in Section 4) with $k = 3$, and Model 3 (that is, the iterative LP-rounding approach discussed in Section 5). All optimization models have been solved by using a branch-and-cut approach implemented in the mathematical programming solver Gurobi 6.50 [1] on a PC with Intel i7-4770 CPU at 3.4 GHz and 16Gb RAM. Collectively, the experiments consist of 5200 instances of the task partitioning problem.

6.2 Discussion of the results

6.2.1 Schedulability

Experiment 1: Variation of $\bar{U}$ (Figure 1) In the first type of experiment, we use the average load $\bar{U}$ as the independent variable. We vary $\bar{U}$ from 0.2 to 1.5. We fix $m = 10$, $\kappa = 10$, $p = 0.5$, $\alpha = 0.2$. We generate 30 task sets for each value of $\bar{U}$. When the percentage of schedulable task sets was strictly between 0 and 1, we generated 20 additional task sets to achieve a higher precision. Figure 1 shows the dependency on $\bar{U}$ of the percentage of task sets that are guaranteed to be schedulable by Corollary 2 (for Model 1), Theorem 3 (for Model 2) and Theorem 6 (for Model 3). Indeed, as could be expected, schedulability decreases when the load factor $\bar{U}$ increases. It is interesting to note that for Model 2, more than 90% of the generated task sets are schedulable as long as $\bar{U} \leq 1$.

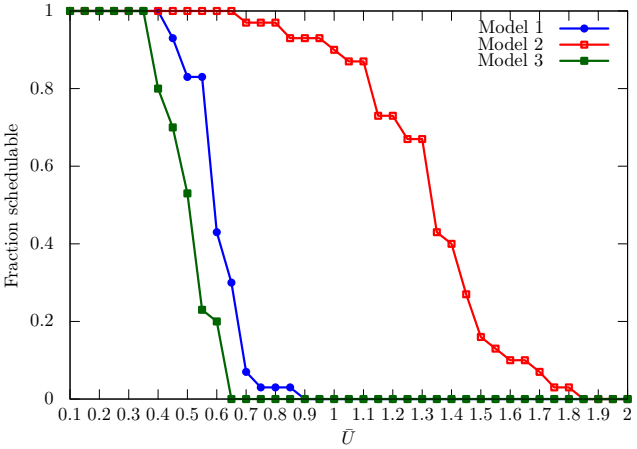

Fig. 1 Experimental results 1: percentage of task sets found to be schedulable by the three models. Variation of $\bar{U}$ ($m = 10$, $\kappa = 10$, $p = 0.5$, $\alpha = 0.2$).

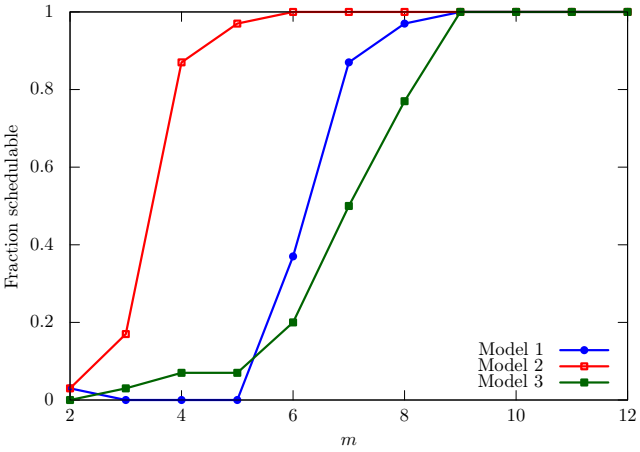

Fig. 2 Experimental results 2: percentage of task sets found to be schedulable by the three models. Variation of m ($\kappa = m$, $\bar{U} = 1$, $p = 0.8$, $\alpha = 0.2$).

Experiment 2: Variation of m (Figure 2) In the second type of experiment, we look at the impact of the number of processors on schedulability, so we vary m from 2 to

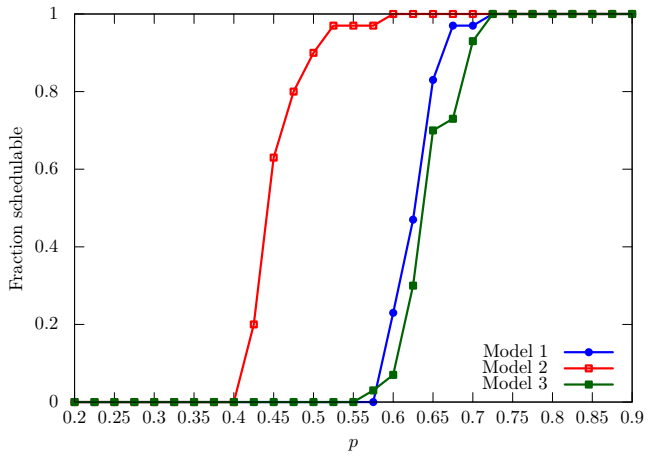

Fig. 3 Experimental results 3: percentage of task sets found to be schedulable by the three models. Variation of p ($m = 10$, $\kappa = 10$, $\bar{U} = 1$, $\alpha = 0.2$).

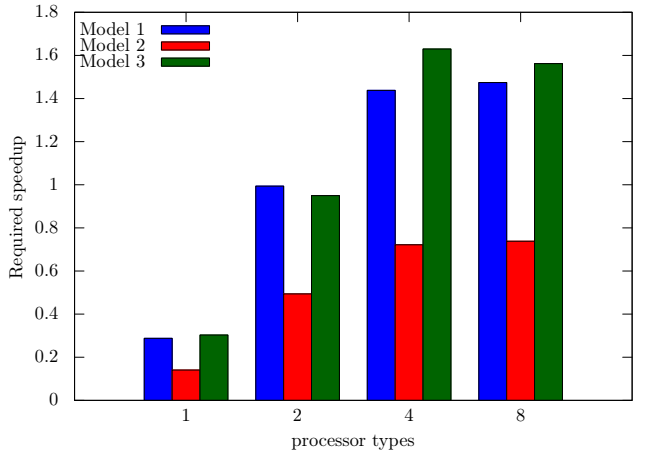

Fig. 4 Experimental results 4: speedup required to guarantee schedulability by the three models. Variation of the number of types of processors ($m = 8$, $\kappa = 8$, $\bar{U} = 0.6$, $p = 0.5$, $\alpha = 0.2$).

12. We fix $\kappa = m$, so $n = m^2$. In this set of experiments, we fix $\bar{U} = 1$, $p = 0.8$, $\alpha = 0.2$. For each experiment, we generated 30 task sets for each value of m. In all cases, schedulability increases with the number of processors even though $\bar{U}$ is fixed. This is due to the fact that, for a fixed total utilization value of each group, a higher number of tasks in a group implies lower utilization values for the individual tasks, and therefore a more efficient partitioning. The best quality is again obtained by Model 2, while Model 1 and Model 3 achieve a similar solution quality.

Experiment 3: Variation of p (Figure 3) In the third type of experiment, we control the sparsity of the processor affinity matrix R, by varying p from 0.2 to 0.9. We fix $m = 10$, $\kappa = 10$, $\bar{U} = 1$, $\alpha = 0.2$. We generate 30 task sets for each value of p. We find out that the sparsity has a high impact on schedulability: there are clear schedulability thresholds around $p = 0.7$ (for Models 1 and 3) and $p = 0.45$ (for Model 2). This is not entirely unexpected, as when the affinity matrix is sparser, it may happen that several tasks of large combined utilization can only be assigned to a small set of processors.

Experiment 4: Variation of the number of processor types (Figure 4) In the fourth type of experiment, we control the effect of similarity among processors. We group the processors into types; we ensure that $c_{i,\tau} = c_{i',\tau}$ whenever i and i' are of the same type. We then vary the number of types of processors. We fix $m = 8$ and for simplicity we consider equal-sized groups, so we vary the number of types in the set $\{1, 2, 4, 8\}$. We also fix $\kappa = 8$, $\bar{U} = 0.6$, $p = 0.5$, $\alpha = 0.2$, and we generate 30 task sets for each value of the independent variable. In this case, it seems more significant to plot the average speedup that is required to ensure schedulability (thus, lower values are preferable, and values below 1 ensure schedulability without speedup). As expected, similarity is helpful, i.e., an increase in the number of types is associated with an increase in the speedup required, particularly for Models 1 and 3.

6.2.2 Running times

In this subsection we discuss the solution time required to solve the instances for the experiments described in the previous section. These times are reported in the Tables 2, 3, 4, and 5. As a general observation, running times for solving Models 1 and 3 are considerably shorter than those needed for Model 2, but as we have seen the latter one is clearly more effective in terms of schedulability. We also note that, for Models 1 and 2, each set of instances generated with the same parameters may have rather variable solution times: the maximum time is often more than 10 times the average time. Model 3, on the other hand, is extremely fast and its maximum times are quite close to the averages.

For Experiment 1, where we vary the average load $\bar{U}$ (Table 2), we observe a slight decrease of the running times as the instances become less schedulable. This is likely due to the solver being able to rule out quickly those instances that happen to be markedly overloaded.

For Experiment 2, where we vary the number m of processors (Table 3), the running times increase rapidly with m, but this is in large part due to the fact that in this scenario we generate $n = m^2$ tasks per instance, so even the size of the input becomes of the order of m^3.

Table 2 Running times for Experiment 1: average and maximum (in secs.) for different ranges of $\bar{U}$.

Model		Range of $\bar{U}$			
		0.1–0.55	0.6–1.05	1.1–1.55	1.6–2.0
Model 1	Mean	0.39	0.37	0.36	0.39
	Max	5.66	2.20	1.45	2.01
Model 2	Mean	9.65	9.73	7.09	4.65
	Max	88.16	89.25	51.72	22.82
Model 3	Mean	0.03	0.03	0.03	0.03
	Max	0.09	0.09	0.12	0.09

Table 3 Running times for Experiment 2: average and maximum (in secs.) for different values of m.

Model		Range of m			
		2–8	9–10	11	12
Model 1	Mean	0.04	0.53	1.65	3.79
	Max	0.26	3.03	13.60	28.50
Model 2	Mean	0.53	18.26	97.68	310.55
	Max	4.47	160.49	380.05	709.27
Model 3	Mean	0.01	0.03	0.04	0.04
	Max	0.04	0.04	0.05	0.07

After discounting this fact, the running times are still positively correlated with m, in particular for Model 2.

For Experiment 3, where we vary the affinity probability p (Table 4), we observe a sharp transition from unschedulable to schedulable instances, and running times progressively increase. Again, this is likely due to the solver being able to rule out quickly those instances where the affinity relation is too sparse to allow schedulability.

For Experiment 4, where we vary the number of types of processors (Table 5), all running time are limited to a few seconds at most and we do not find any significant correlation between the running time and the number of types.

7 Summary and conclusions

In this work, we proposed a partitioning approach for constrained-deadline tasks on heterogenous (unrelated) processors. The approach is based on integer linear programming formulations and allows the derivation of guaranteed speedup bounds and consequently, sufficient schedulability tests.

Experiments using randomly generated task workloads clearly show that the approach based on the solution of Model 2 is viable in terms of computation time, and quite effective in its scheduling capacity, especially when the task-processor affinity relation is dense. On the other hand, the approach based on the solution of Model 1, despite its reduced scheduling capacity, may be of use in the case of instances so large that they cannot be solved with the previous approach in reasonable times. The LP-rounding variant, Model 3, achieves a comparable quality to Model 1 and appears to be extremely fast in practice.

Work supported by NSF grants CNS 1115284, CNS 1218693, CNS 1409175, and CPS 1446631, AFOSR grant FA9550-14-1-0161, ARO grant W911NF-14-1-0499, and a grant from General Motors Corp. This work has also been partially supported by the research project "Designing Human-Agent Collectives for Sustainable Future Societies" (C26A15TXCF) of Sapienza University of Rome, by MIUR PRIN grant 2012JXB3YF_003, and by the National Group of Computing Science (GNCS-INDAM).

References

1. Gurobi Optimizer. `http://www.gurobi.com`.
2. D. Achlioptas, A. Coja-Oghlan, and F. Ricci-Tersenghi. On the solution-space geometry of random constraint satisfaction problems. *Random Structures & Algorithms*, 38(3):251–268 (2011)
3. K. Albers and F. Slomka. An event stream driven approximation for the analysis of real-time systems. In *Proc. 16th Euromicro Conf. on Real-Time Systems*, pages 187–195, IEEE (2004)
4. B. Andersson and G. Raravi. Scheduling constrained-deadline parallel tasks on two-type heterogeneous multiprocessors. In *Proc. 24th Int. Conf. on Real-Time Networks*, pages 247–256, ACM (2016)
5. S. Baruah and E. Bini. Partitioned scheduling of sporadic task systems: an ILP-based approach. In *Proc. 2008 Conference on Design and Architectures for Signal and Image Processing* (2008)
6. S. Baruah, A. Mok, and L. Rosier. Preemptively scheduling hard-real-time sporadic tasks on one processor. In *Proc. 11th Real-Time Systems Symposium*, pages 182–190, IEEE (1990)
7. S. K. Baruah and N. Fisher. The partitioned multiprocessor scheduling of deadline-constrained sporadic task systems. *IEEE Transactions on Computers*, 55(7):918–923 (2006)
8. E. Bini and G. C. Buttazzo. Measuring the performance of schedulability tests. *Real-Time Systems*, 30(1-2):129–154 (2005)
9. V. Bonifaci, G. D'Angelo, and A. Marchetti-Spaccamela. Algorithms for hierarchical and semi-partitioned machine

Table 4 Running times for Experiment 3: average and maximum (in secs.) for different ranges of p.

Model		Range of p			
		0.2–0.375	0.4–0.575	0.6–0.775	0.8–0.925
Model 1	Mean	0.13	0.39	0.55	0.64
	Max	0.70	8.54	2.76	3.81
Model 2	Mean	2.05	8.89	19.66	27.38
	Max	16.91	87.16	138.92	180.02
Model 3	Mean	0.03	0.03	0.03	0.03
	Max	0.04	0.06	0.06	0.04

Table 5 Running times for Experiment 4: average and maximum (in secs.) for different number of types of processors.

Model		Types of processors			
		1	2	4	8
Model 1	Mean	0.16	0.09	0.05	0.07
	Max	0.36	0.22	0.17	0.14
Model 2	Mean	3.14	1.13	0.67	1.06
	Max	11.83	3.54	2.12	4.57
Model 3	Mean	0.02	0.02	0.02	0.02
	Max	0.03	0.03	0.03	0.03

scheduling. To appear in *31st Int. Parallel and Distributed Processing Symposium (IPDPS 2017)*, IEEE.

10. J. Chen and S. Chakraborty. Resource augmentation for uniprocessor and multiprocessor partitioned scheduling of sporadic real-time tasks. *Real-Time Systems*, 49(4):475–516 (2013)
11. H. S. Chwa, J. Seo, J. Lee, and I. Shin. Optimal real-time scheduling on two-type heterogeneous multicore platforms. In *Proc. Real-Time Systems Symposium*, IEEE (2015)
12. M. Dertouzos. Control robotics : the procedural control of physical processors. In *Proceedings of the IFIP Congress*, pages 807–813 (1974)
13. A. Frank and É. Tardos. An application of simultaneous diophantine approximation in combinatorial optimization. *Combinatorica*, 7(1):49–65 (1987)
14. E. C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32(4):755–761 (1985)
15. S. Kamath. Unrelated parallel machine scheduling—perspectives and progress. *OPSEARCH*, 48(4):318–334 (2011)
16. R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, New York (1972)
17. R. M. Karp, F. T. Leighton, R. L. Rivest, C. D. Thompson, U. V. Vazirani, and V. V. Vazirani. Global wire routing in two-dimensional arrays. *Algorithmica*, 2:113–129 (1987)
18. L. C. Lau, R. Ravi, and M. Singh. *Iterative Methods in Combinatorial Optimization*. Cambridge University Press (2011)
19. J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271 (1990)
20. C. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment. *Journal of the ACM*, 20(1):46–61 (1973)
21. A. Marchetti-Spaccamela, C. Rutten, S. van der Ster, and A. Wiese. Assigning sporadic tasks to unrelated machines. *Mathematical Programming*, pages 1–28 (2014)
22. G. Raravi. *Real-Time Scheduling on Heterogeneous Multiprocessors*. PhD thesis, Technical Institute of Porto (Portugal) 2014.
23. G. Raravi, B. Andersson, and K. Bletsas. Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors. *Real-Time Systems*, 49(1):29–72 (2013)
24. G. Raravi, B. Andersson, V. Nélis, and K. Bletsas. Task assignment algorithms for two-type heterogeneous multiprocessors. *Real-Time Systems*, 50(1):87–141 (2013)
25. G. Raravi and V. Nélis. A PTAS for assigning sporadic tasks on two-type heterogeneous multiprocessors. In *Proc. 33rd Real-Time Systems Symposium*, pages 117–126, IEEE (2012)
26. D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474 (1993)
27. A. Wiese, V. Bonifaci, and S. Baruah. Partitioned EDF scheduling on a few types of unrelated multiprocessors. *Real-Time Systems*, 49(2):219–238 (2013)