CISTER
Research Center in
Real-Time & Embedded
Computing Systems

Technical Report

Provably Good Task Assignment for Two-type Heterogeneous Multiprocessors using Cutting Planes

Björn Andersson
Gurulingesh Raravi

CISTER-TR-140511
Version:
Date: 1/1/2014

Provably Good Task Assignment for Two-type Heterogeneous Multiprocessors using Cutting Planes

Björn Andersson, Gurulingesh Raravi

CISTER Research Unit

Polytechnic Institute of Porto (ISEP-IPP)

Rua Dr. António Bernardino de Almeida, 431

4200-072 Porto

Portugal

Tel.: +351.22.8340509, Fax: +351.22.8340509

E-mail: baa@isep.ipp.pt, guhri@isep.ipp.pt

http://www.cister.isep.ipp.pt

Abstract

Consider scheduling of real-time tasks on a multiprocessor where migration is forbidden. Specifically, consider the problem of determining a task-to-processor assignment for a given collection of implicit-deadline sporadic tasks upon a multiprocessor platform in which there are two distinct types of processors. For this problem, we propose a new algorithm, LPC (task-assignment-based-on-solving a Linear Program with Cutting planes). The algorithm offers the following guarantee: for a given task set and a platform, if there exists a feasible task-to-processor assignment, then LPC succeeds in finding such a feasible task-to-processor assignment as well but on a platform in which each processor is 1.5 times faster and has 3 additional processors. For systems with large number of processors, LPC has a better approximation ratio than state-of-the-art algorithms. To the best of our knowledge, this is the first work that develops a provably good real-time task assignment algorithm using cutting planes.

© CISTER Research Unit
www.cister.isep.ipp.pt

A

Provably Good Task Assignment for Two-type Heterogeneous Multiprocessors using Cutting Planes

(Submitted to Special Issue on Real-Time, Embedded and Cyber-Physical Systems)

Björn Andersson, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, USA
Gurulingesh Raravi, CISTER/INESC-TEC, ISEP, Polytechnic Institute of Porto, Porto, Portugal

Consider scheduling of real-time tasks on a multiprocessor where migration is forbidden. Specifically, consider the problem of determining a task-to-processor assignment for a given collection of implicit-deadline sporadic tasks upon a multiprocessor platform in which there are two distinct types of processors. For this problem, we propose a new algorithm, LPC (task-assignment-based-on-solving a Linear Program with Cutting planes). The algorithm offers the following guarantee: for a given task set and a platform, if there exists a feasible task-to-processor assignment, then LPC succeeds in finding such a feasible task-to-processor assignment as well but on a platform in which each processor is 1.5 times faster and has 3 additional processors. For systems with large number of processors, LPC has a better approximation ratio than state-of-the-art algorithms. To the best of our knowledge, this is the first work that develops a provably good real-time task assignment algorithm using cutting planes.

Categories and Subject Descriptors: D.4.7 [**Operating Systems**]: Organization and Design—*Real-time systems and embedded systems*; G.4 [**Mathematical Software**]: Algorithm design and analysis

General Terms: Algorithms, Performance, Theory

Additional Key Words and Phrases: Cutting planes, Heterogeneous multiprocessors, Linear programming, Real-time scheduling

1. INTRODUCTION

This paper addresses the problem of finding an assignment of real-time tasks to processors (also referred to as *partitioning* or *task-to-processor* assignment or *non-migrative* assignment) for a given set of tasks on a heterogeneous multiprocessor platform. We consider *implicit-deadline sporadic tasks*, that is, a task generates a (potentially infinite) sequence of jobs where each job has an execution time and a deadline and for each task, the deadline of a job this task is equal to the minimum time between job arrivals of this task. Such tasks can be used to model a range of applications where the software needs to perform an operation repeatedly on incoming or sampled data, e.g. feedback control systems, signal processing or multimedia playout. We consider a heterogeneous multiprocessor platform comprising processors of two unrelated types: type-1 and type-2 and we refer to such a computing platform as *two-type platform*. Our interest in considering such a platform model is motivated by the fact that many chip makers offer chips having two types of processors [AMD Inc. 2012; Apple Inc. 2012; Intel Corp. 2013b; 2013c; Nvidia Inc. 2013; Qualcomm Inc 2013; Samsung Inc. 2013; Texas Instruments 2012; Alben 2013; Intel Corp. 2013a].

In the partitioning problem, every task must be statically assigned to a processor before run time and all its jobs must execute on that processor at run time (i.e., jobs *cannot* migrate between different processors). The challenge is to find, before run time, a task-to-processor assignment such that, at run time, a uniprocessor scheduling algorithm running on each processor meets all the deadlines of the tasks on the respective processor. Scheduling the tasks to meet deadlines on a uniprocessor platform is a well-understood problem. One may use Earliest-Deadline First (EDF) [Liu and Layland 1973], for example. EDF is an *optimal* scheduling algorithm on a uniprocessor system [Liu and Layland 1973; Dertouzos 1974], with the interpretation that for every valid arrival pattern, if a schedule exists that meets deadlines then EDF constructs a schedule that meets deadlines as well. Therefore, assuming that an optimal schedul-

Table I. Summary of state-of-the-art task assignment algorithms along with the algorithm proposed in this paper.

Computing Platform	Adversary Task migration	Task Assignment Algorithms			
		Algorithm	Task migration	Approx. ratio	Complexity
t-type[a]	non-migrative	[Baruah 2004a]	non-migrative	$2x$	polynomial
t-type	non-migrative	[Baruah 2004b]	non-migrative	$2x$	polynomial
t-type	non-migrative	[Lenstra et al. 1990]	non-migrative	$2x$	polynomial
t-type	fully-migrative	[Correa et al. 2012]	non-migrative	$4x$	polynomial
2-type[b]	non-migrative	[Raravi et al. 2013]	non-migrative	$2x$	polynomial
2-type	intra-migrative	[Raravi et al. 2012]	non-migrative	$2x$	polynomial
t-type	non-migrative	[Horowitz and Sahni 1976]	non-migrative	PTAS[c]	exponential in processors
t-type	non-migrative	[Jansen and Porkolab 1999]	non-migrative	PTAS	exponential in processors
2-type	non-migrative	[Raravi and Nélis 2012]	non-migrative	PTAS	exponential in $1/\epsilon$
t-type	non-migrative	[Wiese et al. 2013]	non-migrative	PTAS	exponential in $1/\epsilon$
2-type	non-migrative	LPC	non-migrative	$1.5x$ and 3 extra processors	polynomial

[a] A heterogeneous multiprocessor platform having two or more processor types.
[b] A heterogeneous multiprocessor platform having only two processor types.
[c] A PTAS takes an instance of an optimization problem and a parameter $\epsilon > 0$ as inputs and, in time polynomial in the problem size (although not necessarily in the value of ϵ), produces a solution that is within a factor $1 + \epsilon$ of being optimal.

ing algorithm is used on each processor, the challenging part is to find a partitioning for which *there exists* a schedule that meets all the deadlines — such a partitioning is said to be a *feasible* partitioning hereafter. Even in the simpler case of identical multiprocessors, finding a feasible partitioning is NP-Complete in the strong sense [Korte and Vygen 2006]. Hence, this result continues to hold for two-type platforms. Our goal in this work is to design an algorithm for assigning tasks to processors on two-type heterogeneous multiprocessors and prove its performance.

In this work, the *resource augmentation* framework [Phillips et al. 1997] is used to characterize the performance of the algorithm under design. We define the *approximation ratio* $\mathrm{AR}_{\mathcal{A}}$ of a (non-migrative) algorithm $\mathcal{A}$ against a (non-migrative) adversary as the lowest number such that for every task set τ and computing platform π it holds that if it is possible for a non-migrative algorithm (i.e., the adversary) to meet all deadlines of τ on π then algorithm $\mathcal{A}$ outputs a task-to-processor assignment which meets all deadlines of τ on a platform π' (when scheduled using uniprocessor EDF [Liu and Layland 1973]) whose every processor is $\mathrm{AR}_{\mathcal{A}}$ times faster than the corresponding processor in π. A low approximation ratio indicates high performance; the best achievable is 1 (which reflects the optimal algorithm for a given problem). Therefore, we aim to design an algorithm with a finite (ideally small) approximation ratio.

Related work. The partitioning problem on heterogeneous multiprocessors has been studied in the past [Baruah 2004b; 2004a; Raravi et al. 2012; Raravi et al. 2013; Raravi and Nélis 2012; Wiese et al. 2013]. It is a well-known fact that the problem under consideration is equivalent to the problem of scheduling a set of non-real-time jobs, arriving at time zero, on unrelated parallel machine, so that they all finish before a specified time. This (equivalent) problem has been studied in [Horowitz and Sahni 1976; Lenstra et al. 1990; Jansen and Porkolab 1999; Correa et al. 2012]. In [Baruah 2004b; 2004a; Lenstra et al. 1990], the authors proposed algorithms for the problem of partitioning implicit-deadline sporadic task sets on heterogeneous multiprocessors with an approximation ratio of 2. All these approaches [Baruah 2004b; 2004a; Lenstra et al. 1990] focused on generic heterogeneous multiprocessor platforms with two or more processor types. Due to practical relevance, recent research [Raravi et al. 2013] considered the partitioning problem on two-type platforms and proposed an algorithm, FF-3C, and couple of its variants based on first-fit heuristic. These had the same per-

formance guarantee as the approaches in [Baruah 2004b; 2004a; Lenstra et al. 1990] (i.e., requiring processors twice as fast, in the worst-case) but can be implemented efficiently and exhibit better average-case performance than those in [Baruah 2004b; 2004a].

Moving to algorithms whose performance has been evaluated against a more powerful adversary, recently, in [Raravi et al. 2012], it is shown that, for the given task set on a two-type platform, if there exists a feasible task-to-processor-type assignment (i.e., tasks are assigned to processor *types* and the jobs *can* migrate between processors of the *same type*) then, the algorithm proposed in [Raravi et al. 2012] succeeds in finding a task-to-processor assignment for the given task set on a platform in which the speed of each processor is twice faster. In [Correa et al. 2012], it is shown that if a task set can be scheduled by an optimal algorithm on a heterogeneous platform with full migrations (i.e., jobs *can* migrate between processors of *any type*) then, an optimal algorithm for scheduling tasks on a heterogeneous platform with no migrations (i.e., non-migrative assignment) needs processors four times as fast.

In [Horowitz and Sahni 1976; Jansen and Porkolab 1999; Raravi and Nélis 2012; Wiese et al. 2013], authors proposed *polynomial-time approximation scheme*s (PTAS) for this problem. A PTAS takes an instance of an optimization problem and a parameter $\epsilon > 0$ as inputs and, in time polynomial in the problem size (although not necessarily in the value of ϵ), produces a solution that is within a factor $1 + \epsilon$ of being optimal. PTAS is theoretically a significant result since such algorithms partition the task set in polynomial time, to any desired degree of accuracy. However, (most often) their practical significance is severely limited due to a very high run-time complexity that they incur.

The state-of-the-art (along with the contributions of this paper) is summarized in Table I. Each row in the table corresponds to a different algorithm. For example, the first row in the table is read as follows: for a generic heterogeneous multiprocessor platform in which there can be two or more types of processors (denoted as t-type), a non-migrative algorithm is proposed in [Baruah 2004a] and this algorithm has an approximation ratio of 2 against a non-migrative adversary and the algorithm has a polynomial time-complexity.

Contribution and Significance of this work. We present an algorithm, LPC (task assignment based on solving a **L**inear **P**rogram with **C**utting planes), for the problem of partitioning a given implicit-deadline sporadic task set on a two-type heterogeneous multiprocessor platform which offers the following guarantee. If there exists a feasible partitioning of a task set τ on a two-type platform π then, LPC succeeds in finding a feasible partitioning of τ as well but on a platform $\pi^{(1.5x+3p)}$ in which each processor is 1.5 times faster than the corresponding processor in π and has 3 additional processors than π.

The significance of this work is two-fold. First, for the problem of non-migrative task assignment, our algorithm, has superior performance compared to state-of-the-art. This can be seen from Table I since, for systems with large number of processors, our algorithm offers a better approximation ratio than all the previous algorithms. This is because (i) for systems with large number of processors, the additional 3 processors that our algorithm requires become negligible and hence its approximation ratio tends to $1.5x$ which is better than the algorithms in [Baruah 2004a; 2004b; Lenstra et al. 1990; Correa et al. 2012; Raravi et al. 2013; Raravi et al. 2012] and (ii) compared to PTAS algorithms [Horowitz and Sahni 1976; Jansen and Porkolab 1999; Raravi and Nélis 2012; Wiese et al. 2013] which incur a very high time-complexity (i.e., exponential in processors or exponential in $1/\epsilon$), our algorithm offers a lower (i.e., polynomial) time-complexity. Second, although task assignment schemes with provably good per-

formance have previously been developed by relaxing an Mixed Integer-Linear Program (MILP) to a Linear Program (LP) (e.g., [Baruah 2004b; 2004a; Lenstra et al. 1990]) and cutting planes have been used to solve (M)ILP in different efforts, no work in the past has shown how cutting planes can be used to improve the approximation ratio of algorithms for provably good algorithms for assigning real-time tasks to processors. Hence, to the best of our knowledge, this work is the first to show how cutting planes can be used to improve the approximation ratio of algorithms for provably good algorithms for assigning real-time tasks to processors.

Organization of the paper. The rest of the paper is organized as follows. Section 2 briefs the system model. Section 3 discusses task assignment using Integer Linear Program, Linear Program relaxation and cutting planes. Section 4 presents our new algorithm, LPC and Section 5 derives its performance. Finally, Section 6 concludes.

2. SYSTEM MODEL

We consider the problem of scheduling a task set $\tau = \{\tau_1, \tau_2, \ldots, \tau_n\}$ of n implicit-deadline sporadic tasks on a two-type heterogeneous multiprocessor platform $\pi = \{\pi_1, \pi_2, \ldots, \pi_m\}$ comprising m processors, of which $|P^t(\pi)|$ are of type-t; where $t \in \{1, 2\}$. The set of processors of type-t is represented by $P^t(\pi)$. Note that $P^1(\pi) \bigcup P^2(\pi) = \pi$. Each task τ_i is characterized by a *worst-case execution time* (WCET) and a *minimum inter-arrival time* T_i (which is equal to its *deadline*). Each task τ_i releases a (potentially infinite) sequence of *jobs*, with the first job released at any time and subsequent jobs released *at least* T_i time units apart. Each job released by a task τ_i has to complete its execution within T_i time units from its release. We assume that an optimal scheduling algorithm (such as EDF [Liu and Layland 1973]) is used to schedule the tasks on each processor.

On a two-type platform, the WCET of a task depends on the type of processor on which the task executes. We denote by $C_{i,1}$ and $C_{i,2}$ the WCET of task τ_i when executed on a processor of type-1 and type-2 and we denote by $u_{i,1} \stackrel{\text{def}}{=} \frac{C_{i,1}}{T_i}$ and $u_{i,2} \stackrel{\text{def}}{=} \frac{C_{i,2}}{T_i}$ the utilizations of task τ_i on type-1 and type-2 processors, respectively. A task that cannot be executed upon a certain processor type is modeled by setting its utilization on that processor type to ∞[1].

We now define a couple of auxiliary functions that are used in the rest of the paper.

Let $\mathrm{aot}\left(\mathrm{ts}: \texttt{ set of tasks},t: \texttt{ type}\right)$ be a function that returns the subset of tasks in ts such that $u_{i,t} > 1/3$. Similarly, let $\mathrm{ah}(\mathrm{ts}, t)$ be a function which returns the subset of tasks in ts such that $u_{i,t} > 1/2$.

Let $\texttt{solve}\left(\mathrm{lp} \texttt{ : linear program}\right)$ be a function which solves the linear program lp and if this solution is not a vertex optimal solution then it converts this solution into a vertex optimal solution (previous work [Baruah 2004b] did such a transformation). It returns the values assigned to variables and the value of the objective function.

Let $\mathrm{mp}\left(\left|P^1\left(\mathrm{pl}\right)\right| \texttt{ : number of processors, } \left|P^2\left(\mathrm{pl}\right)\right| \texttt{ : number of processors, } s\texttt{: relative speed of processors, pl: two-type platform}\right)$ denote a function that returns a computing platform with $|P^1(\mathrm{pl})|$ (resp., $|P^2(\mathrm{pl})|$) processors of type-1 (resp., type-2) that are $s > 0$ times as fast as the corresponding processors of type-1 (resp., type-2) in computing platform pl. Intuitively, "mp" means "make platform". This function is never called by our algorithm; it is only used in proofs.

[1]Later in the paper, we will solve LPs and MILPs and unfortunately, solvers for these problems typically do not allow coefficients to be ∞. This can be dealt with, however, by assigning utilization of a task on a certain processor to $\max(|P^1(\pi)|, |P^2(\pi)|)$. We will see, later in the paper, that this gives the same result as assigning ∞.

Minimize Z_{MILP} subject to the following constraints:

I1.	$\forall p \in P^1\,(\mathrm{pl}) : \sum_{\tau_i \in \mathrm{ts}} xv_{i,p} \times u_{i,1} \leq Z_{\mathrm{MILP}}$
I2.	$\forall p \in P^2\,(\mathrm{pl}) : \sum_{\tau_i \in \mathrm{ts}} xv_{i,p} \times u_{i,2} \leq Z_{\mathrm{MILP}}$
I3.	$\forall \tau_i \in \mathrm{ts} : \sum_{p \in P^1(\mathrm{pl})} xv_{i,p} + \sum_{p \in P^2(\mathrm{pl})} xv_{i,p} = 1$
I4.	$\forall \tau_i \in \mathrm{ts}$ and $\forall p \in P^1\,(\mathrm{pl}) : xv_{i,p}$ is an integer $\in \{0,1\}$
I5.	$\forall \tau_i \in \mathrm{ts}$ and $\forall p \in P^2\,(\mathrm{pl}) : xv_{i,p}$ is an integer $\in \{0,1\}$

Fig. 1. $\mathrm{MILP}_{\mathrm{OPT}}(\mathrm{ts}, \mathrm{pl})$– MILP formulation for assigning tasks in ts to processors in pl.

Let $\mathrm{sched}(A, \tau, \pi)$ denote a predicate to signify that the task-to-processor assignment returned by algorithm A for tasks in τ onto processors in π *meets all the deadlines* when the tasks assigned to each processor are scheduled by an optimal uniprocessor scheduling algorithm (such as EDF [Liu and Layland 1973]). The term *meets all the deadlines* in this and other predicates means 'meets deadlines for every possible arrival of tasks that is valid as per the given parameters of τ'. The predicates with $A = \mathrm{OPT}$ imply that there exists a feasible task-to-processor assignment of tasks in τ onto processors in π.

3. TASK ASSIGNMENT, MILP, LP AND CUTTING PLANES

In this section, we describe how the task assignment problem under consideration can be formulated as MILP. Recall that we mentioned in Section 1 that given a task set and a computer platform, the problem of deciding if a feasible task assignment exists is NP-complete in the strong sense. Then it clearly follows that, for any MILP formulation of this problem, deciding if the MILP is feasible is NP-complete in the strong sense as well. Since deciding if our MILP formulation of task assignment is NP-complete, we also discuss how it can be relaxed to LP (because LP can be solved in polynomial time).

Recall that, once the tasks are assigned to processors (also referred to as *task-to-processor* or *non-migrative* assignment or *partitioning*), we assume that an optimal scheduling algorithm (such as EDF [Liu and Layland 1973]) is used on each processor to schedule the respective tasks. From the uniprocessor feasibility test, the following necessary and sufficient condition must hold $\forall t \in \{1, 2\}$ in order for the non-migrative task assignment to be feasible:

$$\forall \pi_p \in P^t(\pi) : \quad \sum_{\tau_i \in \tau[\pi_p]} u_{i,t} \;\leq\; 1 \tag{1}$$

where $\tau[\pi_p]$ denotes the tasks assigned to processor $\pi_p \in \pi$.

The problem of assigning tasks in τ to processors in π can be formulated as MILP using the function $\mathrm{MILP}_{\mathrm{OPT}}(\tau, \pi)$ which returns an MILP formulation as defined by Figure 1. In this MILP formulation, the indicator variable $xv_{i,p}$ indicates the assignment of task τ_i to processor π_p, i.e., $xv_{i,p} = 1$ implies that τ_i is entirely assigned to processor π_p, $xv_{i,p} = 0$ implies that τ_i is not assigned to processor π_p. The variable Z_{MILP} denotes the maximum capacity of any processor that is used and is set as the objective function (to be minimized). If $Z_{\mathrm{MILP}} \leq 1$ then it implies that the sum of utilization of tasks assigned to any processor is less than or equal to the available capacity on that processor and hence the assignment is feasible. If $Z_{\mathrm{MILP}} > 1$ then it implies that the condition in Expression (1) is violated and hence the task set is *non-migrative infeasible*, i.e., no task assignment algorithm will be able to assign the given tasks on the given processors such that all the deadlines are met.

We now illustrate this with an example. Consider a task set $\tau = \{\tau_1, \tau_2, \tau_3, \tau_4\}$ comprising four tasks and a two-type platform $\pi = \{\pi_1, \pi_2, \pi_3\}$ comprising three processors of which π_1 and π_2 are of type-1 and π_3 is of type-2. The utilizations of these tasks

Table II. An example task set.

	Type-1 (π_1, π_2)	**Type-2** (π_3)
Task	$\mathbf{u_{i,1}}$	$\mathbf{u_{i,2}}$
τ_1	0.51	1.1
τ_2	0.51	1.1
τ_3	0.51	1.1
τ_4	1.1	0.5

on type-1 and type-2 processors are shown in Table II. Note that this task set is non-migrative infeasible on the given platform.

Solving the MILP formulation, $\mathrm{MILP}_{\mathrm{OPT}}(\tau, \pi)$, for this example outputs $Z_{\mathrm{MILP}} = 1.02$ (corresponding to the assignment in which τ_1 and τ_2 are assigned to π_1 of type-1, τ_3 is assigned to π_2 of type-1 and τ_4 is assigned to π_3 of type-2). Since $Z_{\mathrm{MILP}} > 1$, it rightly indicates that the task set is non-migrative infeasible on the given platform.

As stated earlier (in Section 1) the problem of finding a feasible task-to-processor assignment on two-type heterogeneous multiprocessors is NP-Complete in the strong sense. Since $\mathrm{MILP}_{\mathrm{OPT}}(\mathrm{ts}, \mathrm{pl})$, shown in Figure 1, is the MILP formulation for this problem, it holds that $\mathrm{MILP}_{\mathrm{OPT}}(\mathrm{ts}, \mathrm{pl})$ is NP-Complete in the strong sense as well. It has been shown in the past that, via relaxation of (M)ILP formulation to LP (by allowing a certain number of tasks to be fractionally assigned to processors initially) and certain *rounding tricks* [Potts 1985] (for integrally assigning the fractionally assigned tasks), polynomial time-complexity can be attained [Baruah 2004a; 2004b; Lenstra et al. 1990] at the expense of potentially non-optimal value for the objective function. Recently, it was shown that assigning tasks to *processor types* first (also referred to as *task-to-processor-type* or *intra-migrative* assignment since jobs are allowed to migrate between processors of the same type) and then assigning them to individual *processors* lead to a better performance [Raravi et al. 2012] than [Baruah 2004a; 2004b; Lenstra et al. 1990]. Hence, in addition to using *cutting planes* in this work, we also use the above mentioned two tricks, i.e., (i) assigning tasks to processor types first and then assigning them to individual processors and (ii) relaxing MILP to LP and then integrally assigning the fractional tasks.

In intra-migrative task assignment, once tasks have been assigned to processor types, we can use an optimal identical multiprocessor scheduling algorithm (e.g., Sporadic-EKG [Andersson and Bletsas 2008] with $s = \gcd(T_1, \ldots, T_n)$, ERfair [Anderson and Srinivasan 2000], DP-WRAP [Levin et al. 2010]) to schedule them on processors of each type. From the feasibility tests of identical multiprocessor scheduling, the following conditions must hold $\forall t \in \{1, 2\}$ in order for intra-migrative task assignment to be feasible:

$$\forall \tau_i \in \tau^t : \quad u_{i,t} \leq 1 \tag{2}$$

$$\sum_{\tau_i \in \tau^t} u_{i,t} \leq \left| P^t(\pi) \right| \tag{3}$$

where τ^t denotes the tasks assigned to processors of type-t. Given these necessary and sufficient feasibility conditions, we now describe how to obtain a task-to-processor-type assignment of τ on π.

We partition the task set τ into four subsets $\mathrm{H12}(\tau, 1)$, $\mathrm{H1}(\tau, 1)$, $\mathrm{H2}(\tau, 1)$ and $\mathrm{L}(\tau, 1)$ as defined below.

$$\mathrm{H12}(\mathrm{ts}, \theta) = \{\tau_i \in \mathrm{ts} : u_{i,1} > \theta \wedge u_{i,2} > \theta\} \tag{4}$$

$$\mathrm{H1}(\mathrm{ts}, \theta) = \{\tau_i \in \mathrm{ts} : u_{i,1} \leq \theta \wedge u_{i,2} > \theta\} \tag{5}$$

$$\mathrm{H2}(\mathrm{ts}, \theta) = \{\tau_i \in \mathrm{ts} : u_{i,1} > \theta \wedge u_{i,2} \leq \theta\} \tag{6}$$

$$\mathrm{L}(\mathrm{ts}, \theta) = \{\tau_i \in \mathrm{ts} : u_{i,1} \leq \theta \wedge u_{i,2} \leq \theta\} \tag{7}$$

Minimize Z subject to the following constraints:

I1.	$U^1 + \sum_{\tau_i \in \mathrm{ts}} yv_{i,1} \times u_{i,1} \leq Z \times \lvert P^1(\mathrm{pl})\rvert$
I2.	$U^2 + \sum_{\tau_i \in \mathrm{ts}} yv_{i,2} \times u_{i,2} \leq Z \times \lvert P^2(\mathrm{pl})\rvert$
I3.	$\forall \tau_i \in \mathrm{ts}$: $yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau_i \in \mathrm{ts}$: $yv_{i,1}$ is an **integer** $\in \{0, 1\}$
I5.	$\forall \tau_i \in \mathrm{ts}$: $yv_{i,2}$ is an **integer** $\in \{0, 1\}$

Fig. 2. $\mathrm{MILP}_{\mathrm{TYPE}}(\mathrm{ts}, \mathrm{pl}, U^1, U^2)$ — MILP formulation for assigning tasks in ts to processor types in pl.

$\mathrm{H12}(\tau, 1)$ is the set of tasks whose utilization exceeds one on both processor types. These tasks cannot be assigned to any of the processor types as assigning them in such a manner violates the condition in Expression (2). Hence, these tasks make the task set infeasible and thus we assume this set to be empty in the rest of this section. $\mathrm{H1}(\tau, 1)$ is the set of tasks that must be assigned to type-1 processors as their utilization on type-2 processors exceeds one and hence assigning them to type-2 processors violates the condition in Expression (2). Analogously, $\mathrm{H2}(\tau, 1)$ is the set of tasks that must be assigned to type-2 processors as their utilization on type-1 processors exceeds one. Finally, $\mathrm{L}(\tau, 1)$ is the set of tasks that can be assigned on either processor type as their utilizations on both processor types do not exceed one. In these definitions, we can intuitively understand the meaning of "H" as "heavy" and "L" as "light" tasks. Now, to obtain an intra-migrative task assignment, do the following.

First, assign the tasks in $\mathrm{H1}(\tau, 1)$ to type-1 (resp., $\mathrm{H2}(\tau, 1)$ to type-2) processors. Let U^1 refer to the capacity used on type-1 processors after assigning $\mathrm{H1}(\tau, 1)$ tasks, i.e., $U^1 = \sum_{\tau_i \in \mathrm{H1}(\tau,1)} u_{i,1}$. Analogously, let $U^2 = \sum_{\tau_i \in \mathrm{H2}(\tau,1)} u_{i,2}$. If $U^1 > \left|P^1(\pi)\right|$ or $U^2 > \left|P^2(\pi)\right|$ then the task set is intra-migrative infeasible as this violates the condition in Expression (3).

Second, solve the formulation, $\mathrm{MILP}_{\mathrm{TYPE}}(\mathrm{L}(\tau, 1), \pi, U^1, U^2)$, of Figure 2 for assigning tasks in $\mathrm{L}(\tau, 1)$. In this formulation, each variable, $yv_{i,t}$ ($t \in \{1, 2\}$), indicates the assignment of task τ_i to type-t processors. The variable Z denotes the average used capacity of either type-1 or type-2 processors, whichever is greater, and is set as the objective function to be minimized. If $Z \leq 1$ then a successful intra-migrative assignment is obtained else the task set is intra-migrative infeasible as it violates Expression (3).

Recall that, we are interested in obtaining a non-migrative (i.e., task-to-processor) assignment. However, this two-step algorithm where the "Heavy" tasks are assigned first and then the "Light" tasks are assigned by solving the MILP formulation (of Figure 2) gives us a task-to-processor-type assignment. Hence, we need to convert this task-to-processor-type assignment into a task-to-processor assignment. However, for some task sets, it may be the case that a feasible task-to-processor-type assignment exists but not a feasible task-to-processor assignment. As a result of this, the two-step algorithm can sometimes indicate that a feasible task-to-processor-type assignment exist for those task sets which do not have a feasible task-to-processor assignment. To illustrate this, let us apply this two-step algorithm on our earlier example (see Table II). It first partitions the tasks as follows: $\mathrm{H1}(\tau, 1) = \{\tau_1, \tau_2, \tau_3\}$ and $\mathrm{H2}(\tau, 1) = \{\tau_4\}$. Then, it assigns all the $\mathrm{H1}(\tau, 1)$ tasks to type-1 processors and $\mathrm{H2}(\tau, 1)$ tasks to type-2 processors. As a result, we obtain: $Z = 0.765$ indicating that a feasible task-to-processor-type assignment exists. But, we cannot convert this assignment into a feasible task-to-processor assignment (since for this task set there is no feasible task-to-processor assignment as illustrated earlier). To avoid such undesirable scenarios, we use *cuts*.

Observe that, for the example under consideration, the problem with the returned task-to-processor-type assignment (considering the fact that this must be converted to a task-to-processor assignment) is that *three* tasks with utilization 0.51 on type-1

Minimize zv subject to the following constraints:

C1.	$\sum_{\tau_i \in \text{ts}} yv_{i,1} \times u_{i,1} + \sum_{\tau_i \in \text{pa1}} u_{i,1} \leq \left\lvert P^1\,(\text{pl})\right\rvert \times zv$
C2.	$\sum_{\tau_i \in \text{ts}} yv_{i,2} \times u_{i,2} + \sum_{\tau_i \in \text{pa2}} u_{i,2} \leq \left\lvert P^2\,(\text{pl})\right\rvert \times zv$
C3.	$\forall \tau_i \in \text{ts} : yv_{i,1} + yv_{i,2} = 1$
C4.	$\sum_{\tau_i \in \text{fun}(\text{ts} \cup \text{pa1},\ 1)} yv_{i,1} \leq \left\lvert P^1\,(\text{pl})\right\rvert$
C5.	$\sum_{\tau_i \in \text{fun}(\text{ts} \cup \text{pa2},\ 2)} yv_{i,2} \leq \left\lvert P^2\,(\text{pl})\right\rvert$
C6.	$\forall \tau_i \in \text{ts}$: $yv_{i,1}$ is a real number ≥ 0
C7.	$\forall \tau_i \in \text{ts}$: $yv_{i,2}$ is a real number ≥ 0

Fig. 3. $\text{TLP}_{\text{CUT}}(\text{ts}, \text{pl}, \text{pa1}, \text{pa2}, \text{fun})$ — LP formulation with *cuts* for assigning tasks in ts to processor types in pl.

processors are assigned to *two* type-1 processors. We know that such an assignment is task-to-processor infeasible as the number of tasks assigned on type-1 processors with their utilizations greater than 0.5 cannot exceed the number of processors of type-1. Analogous property holds for type-2 processors. Hence, we add these two observations as two separate constraints in the MILP formulation (of Figure 2) — these constraints *cut* the feasible region of the optimization problem without losing any solution that is of interest to us (which is a feasible task-to-processor assignment).

Also, as described earlier, solving an MILP formulation is time consuming. However, an LP formulation can be solved in polynomial time though [Karmakar 1984]. So, the MILP formulation for assigning tasks in L is relaxed to an LP formulation to be able to solve it in polynomial-time. This relaxed LP formulation along with the two cuts is obtained by the function $\text{TLP}_{\text{CUT}}\left(\text{L}(\tau, 1), \pi, \text{H1}(\tau, 1), \text{H2}(\tau, 1), \text{ah}\right)$ as shown in Figure 3. In this LP formulation, variables zv and $yv_{i,t}$ have the same meaning as the corresponding variables, Z and $yv_{i,t}$, in the MILP formulation (of Figure 2) and the first three constraints are the same as well. The fourth and fifth constraints represent the cuts that we have added and the sixth and seventh constraints (are *relaxed* versions of fourth and fifth constraints in Figure 2) assert that a task can either be *integrally* or *fractionally* assigned to processor types.

The proposed algorithm which is discussed in the next section uses this Linear Program formulation (which is based on cuts).

4. THE NEW ALGORITHM: LPC

The pseudo-code for the proposed algorithm, LPC, is listed in Algorithm 1. LPC uses a variant of First-Fit bin-packing scheme where heavy tasks are assigned first — pseudo-code for this First-Fit bin-packing variant, FF_{hf}, is shown in Algorithm 2.

The algorithm, LPC, for assigning the tasks in τ to processors in π works as follows.

(1) Partition the task set τ into $\text{H12}(\tau, 2/3)$, $\text{H1}(\tau, 2/3)$, $\text{H2}(\tau, 2/3)$ and $\text{L}(\tau, 2/3)$ as shown in Expression (4)–(7).
(2) Set aside three processors of type-1 (denoted by set rp). Then solve the LP formulation, $\text{TLP}_{\text{CUT}}\left(\text{L}(\tau, 2/3), \pi', \text{H1}(\tau, 2/3), \text{H2}(\tau, 2/3), \text{aot}\right)$, for assigning tasks in $\text{L}(\tau, 2/3)$ to processor types, where $\pi' = \pi \setminus \text{rp}$. In the solution returned by the LP solver, let (i) L1 and L2 denote the subset of tasks in $\text{L}(\tau, 2/3)$ that are integrally assigned to type-1 and type-2 processors, respectively and (ii) τ^F denote the subset of tasks in $\text{L}(\tau, 2/3)$ that are fractionally assigned between processors of type-1 and type-2 (we later show that $|\tau^F| \leq 3$).
(3) Assign the tasks in $\text{H1}(\tau, 2/3) \cup \text{L1}$ to type-1 processors and $\text{H2}(\tau, 2/3) \cup \text{L2}$ to type-2 processors using the First-Fit bin-packing variant, FF_{hf}.
(4) Assign each of the (at most three) tasks in τ^F to a unique processor in rp.

ALGORITHM 1: LPC: The new task assignment algorithm for two-type heterogeneous multiprocessor platforms based on linear program with cuts

Input : A task set τ and a two-type platform π
Output: An assignment of tasks to processors indicated by matrix X
// Let Y denote a matrix in which the algorithm stores the information about the assignment of tasks to processor types
Set each element in X and Y to zero
Select any subset of three processors of type-1 from π and let rp denote this set of processors
Let π' denote a platform $\pi \setminus \mathrm{rp}$
Partition the task set τ into subsets $\mathrm{H12}(\tau, 2/3)$, $\mathrm{H1}(\tau, 2/3)$, $\mathrm{H2}(\tau, 2/3)$ and $\mathrm{L}(\tau, 2/3)$ as shown in Expressions (4)–(7).
if $(\mathrm{H12}(\tau, 2/3) = \emptyset)$ **then**
  foreach $(\tau_\mathrm{i} \in \mathrm{H1}(\tau, 2/3))$ **do** $y_{i,1} := 1$
  foreach $(\tau_\mathrm{i} \in \mathrm{H2}(\tau, 2/3))$ **do** $y_{i,2} := 1$
  $\langle YV, zv, f \rangle :=$ solve$(\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau, 2/3), \pi', \mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot}))$
  if $(\mathrm{f} = \mathrm{feasible})$ **then**
    foreach $(\tau_i \in \mathrm{L}(\tau, 2/3))$ **do** $y_{i,1} := yv_{i,1}$
    foreach $(\tau_i \in \mathrm{L}(\tau, 2/3))$ **do** $y_{i,2} := yv_{i,2}$
    $z := zv$
    if $(z \leq 2/3)$ **then**
      $\tau^{\mathrm{F}} := \{\tau_i \in \mathrm{L}(\tau, 2/3) : y_{i,1} > 0 \ \wedge \ y_{i,2} > 0\}$
      $\tau^{\mathrm{A}} := \mathrm{FF}_{\mathrm{hf}}(\tau^{\mathrm{F}}, \pi, \mathrm{rp}, 1)$
      if $(\tau^{\mathrm{A}} = \tau^{\mathrm{F}})$ **then**
        $\mathrm{L1} := \{\tau_i \in \mathrm{L}(\tau, 2/3) : y_{i,1} = 1\}$ $\quad$ $\mathrm{L2} := \{\tau_i \in \mathrm{L}(\tau, 2/3) : y_{i,2} = 1\}$
        $\tau^1 := \mathrm{L1} \cup \mathrm{H1}(\tau, 2/3)$ $\quad$ $\tau^2 := \mathrm{L2} \cup \mathrm{H2}(\tau, 2/3)$
        if $(\mathrm{aot}(\tau^1, 1) \leq |\mathrm{P}^1(\pi)| - |\mathrm{rp}|)$ **then**
          if $(\mathrm{aot}(\tau^2, 2) \leq |\mathrm{P}^2(\pi)|)$ **then**
            $\tau^{\mathrm{A1}} := \mathrm{FF}_{\mathrm{hf}}(\tau^1, \pi, P^1(\pi) \setminus \mathrm{rp}, 1)$
            $\tau^{\mathrm{A2}} := \mathrm{FF}_{\mathrm{hf}}(\tau^2, \pi, P^2(\pi), 2)$
            if $(\tau^{\mathrm{A1}} = \tau^1)$ **then**
              if $(\tau^{\mathrm{A2}} = \tau^2)$ **then**
                declare SUCCESS
              else
                declare FAILURE
              end
            else
              declare FAILURE
            end
          else
            declare FAILURE
          end
        else
          declare FAILURE
        end
      else
        declare FAILURE
      end
    else
      declare FAILURE
    end
  else
    declare FAILURE
  end
else
  declare FAILURE
end

Informally, choosing $\theta = 2/3$ for partitioning the task set τ into four subsets (Step 1) and then assigning the tasks in $\mathrm{H1}(\tau, 2/3)$ and $\mathrm{H2}(\tau, 2/3)$ to type-1 and type-2 proces-

ALGORITHM 2: $\mathrm{FF_{hf}}$: A variant of First-Fit bin-packing (in which **h**eavy utilization tasks are assigned **f**irst)

Input : ts : a set of tasks, pl : a two-type platform, ps : a set of processors to assign the tasks in ts to, t : type-id
Output: A task-to-processor assignment of tasks in ts to processors in ps of type t of platform pl
// Assumption: $|\mathrm{aot}(\mathrm{ts}, t)| \leq |ps|$
// This algorithm modifies the variable X in the task assignment algorithm, LPC.
// pso is a local variable, a tuple that stores the set of processors in ps in a certain order. The function first(pso) returns the first processor in pso and the function next(pso,p) returns NULL if p is the last processor in pso, otherwise it returns the processor after p in pso. tso is a local variable, a tuple that stores the set of tasks in ts in a certain order.
at $:= \emptyset$ // set 'assigned tasks' to empty
Order the processors in the set ps in some order and assign it to the tuple pso
p := first(pso)
Order the tasks in the set $\mathrm{aot}(\mathrm{ts}, t)$ in some order and assign it to the tuple tso
foreach ($\tau_i \in$ tso), *in order* **do**
    $x_{i,p} := 1$
    at $:=$ at $\cup \{\tau_i\}$
    p := next(pso,p)
    // We will not run out of processors here because of the assumption $|\mathrm{aot}(\mathrm{ts}, t)| \leq |ps|$. Also, note that the main algorithm checks to ensure that when we call this algorithm, this assumption is true
end
Order the tasks in the set ts $\setminus \mathrm{aot}(\mathrm{ts}, t)$ in some order and assign it to the tuple tso
foreach ($\tau_i \in$ tso), *in order* **do**
    p := first(pso)
    while (τ_i *is not in* at) **do**
        if $\left(\sum_{\tau_j \in \mathrm{at}} x_{j,p} \times u_{j,t} + u_{i,t} \leq 1\right)$ **then**
            $x_{i,p} := 1$
            at $:=$ at $\cup \{\tau_i\}$
        else
            if (*next(pso,p)=NULL*) **then**
                return at
            else
                p := next(pso,p)
            end
        end
    end
end
return at

sors, respectively (Step 3), facilitates in creating an algorithm with the desired approximation ratio. Since we will compare the performance of our new algorithm versus every other algorithm that uses processors of at most $2/3$ the speed, it ensures that each of the tasks in $\mathrm{H1}(\tau, 2/3)$ and $\mathrm{H2}(\tau, 2/3)$ is assigned to the same corresponding processor type as under every other successful assignment algorithm. Also, using the function aot while formulating the LP formulation (Step 2) serves the same purpose of achieving the desired approximation ratio — details are provided later in the proofs.

5. THE PERFORMANCE OF THE NEW ALGORITHM

In this section, we show that if there exists a feasible task-to-processor assignment for tasks in task set τ to processors on platform π, then LPC succeeds in finding such a feasible task-to-processor assignment as well for tasks in τ to processors on a platform $\pi^{(1.5x+3p)}$ in which each processor is 1.5 times faster and has 3 additional processors than π. We prove this via a series of intermediate results.

Let $Z_{\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau,2/3),\pi,\mathrm{H1}(\tau,2/3),\mathrm{H2}(\tau,2/3),\mathrm{aot})}$ denote the value of the objective function obtained by solving the LP formulation, $\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau,2/3),\pi,\mathrm{H1}(\tau,2/3),\mathrm{H2}(\tau,2/3),\mathrm{aot})$.

LEMMA 5.1. *Consider a task set τ and a two-type platform π. Let τ' be defined as:*

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \ \wedge \ u'_{i,2} = u_{i,2} \times 3/2$$

It then holds that:

$$\mathrm{sched}(\mathrm{OPT}, \tau', \pi) \Rightarrow Z_{\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau,2/3),\pi,\mathrm{H1}(\tau,2/3),\mathrm{H2}(\tau,2/3),\mathrm{aot})} \leq 2/3$$

PROOF. We assume that the left-hand side predicate is true and show that the right-hand side predicate is true as well. Since $\mathrm{sched}(\mathrm{OPT}, \tau', \pi)$ is true, it holds that:
The value of the objective function for an optimal solution of the following optimization problem is ≤ 1:

Minimize zv subject to the following constraints:

I1.	$\forall p \in P^1(\pi) : \sum_{\tau'_i \in \tau'} xv_{i,p} \times u'_{i,1} \leq zv$
I2.	$\forall p \in P^2(\pi) : \sum_{\tau'_i \in \tau'} xv_{i,p} \times u'_{i,2} \leq zv$
I3.	$\forall \tau'_i \in \tau' : \sum_{p \in P^1(\pi)} xv_{i,p} + \sum_{p \in P^2(\pi)} xv_{i,p} = 1$
I4.	$\forall \tau'_i \in \tau'$ and $\forall p \in P^1(\pi) : xv_{i,p}$ is an integer $\in \{0,1\}$
I5.	$\forall \tau'_i \in \tau'$ and $\forall p \in P^2(\pi) : xv_{i,p}$ is an integer $\in \{0,1\}$

We can observe that there can be at most $\left|P^1(\pi)\right|$ tasks (resp., at most $\left|P^2(\pi)\right|$ tasks) $\tau'_i \in \tau'$ with $u'_{i,1} > 1/2$ (resp., $u'_{i,2} > 1/2$) that are assigned to type-1 processors (resp., type-2 processors). This gives us:
The value of the objective function for an optimal solution of the following optimization problem is ≤ 1:

Minimize zv subject to the following constraints:

I1.	$\forall p \in P^1(\pi) : \sum_{\tau'_i \in \tau'} xv_{i,p} \times u'_{i,1} \leq zv$
I2.	$\forall p \in P^2(\pi) : \sum_{\tau'_i \in \tau'} xv_{i,p} \times u'_{i,2} \leq zv$
I3.	$\forall \tau'_i \in \tau' : \sum_{p \in P^1(\pi)} xv_{i,p} + \sum_{p \in P^2(\pi)} xv_{i,p} = 1$
I4.	$\forall \tau'_i \in \tau'$ and $\forall p \in P^1(\pi) : xv_{i,p}$ is an integer $\in \{0,1\}$
I5.	$\forall \tau'_i \in \tau'$ and $\forall p \in P^2(\pi) : xv_{i,p}$ is an integer $\in \{0,1\}$
I6.	$\sum_{p \in P^1(\pi)} \sum_{\tau'_i \in \tau' : u'_{i,1} > 1/2} xv_{i,p} \leq \left|P^1(\pi)\right|$
I7.	$\sum_{p \in P^2(\pi)} \sum_{\tau'_i \in \tau' : u'_{i,2} > 1/2} xv_{i,p} \leq \left|P^2(\pi)\right|$

Let us rewrite the two last constraints by changing the order of summation on the left-hand side. Also, for each of the first two constraints, let us add up the constraints. This may change the feasible region but the feasible region increases in the sense that each point that was feasible before is still feasible. This gives us:
The value of the objective function for an optimal solution of the following optimization problem is ≤ 1:

Minimize zv subject to the following constraints:

I1.	$\sum_{p \in P^1(\pi)} \sum_{\tau'_i \in \tau'} xv_{i,p} \times u'_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{p \in P^2(\pi)} \sum_{\tau'_i \in \tau'} xv_{i,p} \times u'_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau'_i \in \tau' : \sum_{p \in P^1(\pi)} xv_{i,p} + \sum_{p \in P^2(\pi)} xv_{i,p} = 1$
I4.	$\forall \tau'_i \in \tau'$ and $\forall p \in P^1(\pi) : xv_{i,p}$ is an integer $\in \{0,1\}$
I5.	$\forall \tau'_i \in \tau'$ and $\forall p \in P^2(\pi) : xv_{i,p}$ is an integer $\in \{0,1\}$
I6.	$\sum_{\tau'_i \in \tau' : u'_{i,1} > 1/2} \sum_{p \in P^1(\pi)} xv_{i,p} \leq \left\lvert P^1(\pi) \right\rvert$
I7.	$\sum_{\tau'_i \in \tau' : u'_{i,2} > 1/2} \sum_{p \in P^2(\pi)} xv_{i,p} \leq \left\lvert P^2(\pi) \right\rvert$

Once again, let us reorder the summation on the left-hand side of the two first constraints. Also, extracting the utilization terms outside one of the summations in the first two constraints and then replacing (i) $\sum_{p \in P^1(\pi)} xv_{i,p}$ with $yv_{i,1}$ and (ii) $\sum_{p \in P^2(\pi)} xv_{i,p}$ with $yv_{i,2}$ gives us:
The value of the objective function for an optimal solution of the following optimization problem is ≤ 1:

Minimize zv subject to the following constraints:

I1.	$\sum_{\tau'_i \in \tau'} u'_{i,1} \times yv_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{\tau'_i \in \tau'} u'_{i,2} \times yv_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau'_i \in \tau' : yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau'_i \in \tau'$: $yv_{i,1}$ is an integer $\in \{0,1\}$
I5.	$\forall \tau'_i \in \tau'$: $yv_{i,2}$ is an integer $\in \{0,1\}$
I6.	$\sum_{\tau'_i \in \tau' : u'_{i,1} > 1/2} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
I7.	$\sum_{\tau'_i \in \tau' : u'_{i,2} > 1/2} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

We partition τ' into $\mathrm{H12}(\tau', 1)$, $\mathrm{H1}(\tau', 1)$, $\mathrm{H2}(\tau', 1)$ and $\mathrm{L}(\tau', 1)$ as shown in Expressions (4)–(7). Rewriting based on these partitions gives us:
The value of the objective function for an optimal solution of the following optimization problem is ≤ 1:

Minimize zv subject to the following constraints:

I1.	$\sum_{\tau'_i \in \tau'} u'_{i,1} \times yv_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{\tau'_i \in \tau'} u'_{i,2} \times yv_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau'_i \in \mathrm{H12}(\tau', 1) : yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau'_i \in \mathrm{H1}(\tau', 1) : yv_{i,1} + yv_{i,2} = 1$
I5.	$\forall \tau'_i \in \mathrm{H2}(\tau', 1) : yv_{i,1} + yv_{i,2} = 1$
I6.	$\forall \tau'_i \in \mathrm{L}(\tau', 1) : yv_{i,1} + yv_{i,2} = 1$
I7.	$\forall \tau'_i \in \mathrm{H12}(\tau', 1) : yv_{i,1}, yv_{i,2}$ are integers $\in \{0,1\}$
I8.	$\forall \tau'_i \in \mathrm{H1}(\tau', 1) : yv_{i,1}, yv_{i,2}$ are integers $\in \{0,1\}$
I9.	$\forall \tau'_i \in \mathrm{H2}(\tau', 1) : yv_{i,1}, yv_{i,2}$ are integers $\in \{0,1\}$
I10.	$\forall \tau'_i \in \mathrm{L}(\tau', 1) : yv_{i,1}, yv_{i,2}$ are integers $\in \{0,1\}$
I11.	$\sum_{\tau'_i \in \tau' : u'_{i,1} > 1/2} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
I12.	$\sum_{\tau'_i \in \tau' : u'_{i,2} > 1/2} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

Since $zv \leq 1$, it follows that, $\forall \tau_i \in \mathrm{H1}(\tau', 1)$: $yv_{i,1} = 1$. Analogously, it follows that, $\forall \tau_i \in \mathrm{H2}(\tau', 1)$: $yv_{i,2} = 1$. Also, because $zv \leq 1$, the set $\mathrm{H12}(\tau', 1)$ must be empty. These observations and rearrangement of the terms in the first two constraints gives us:
The value of the objective function for an optimal solution of the following optimization

problem is ≤ 1:

Minimize zv subject to the following constraints:

I1.	$\sum_{\tau'_i \in \mathrm{L}(\tau',1)} u'_{i,1} \times yv_{i,1} + \sum_{\tau'_i \in \mathrm{H1}(\tau',1)} u'_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{\tau'_i \in \mathrm{L}(\tau',1)} u'_{i,2} \times yv_{i,2} + \sum_{\tau'_i \in \mathrm{H2}(\tau',1)} u'_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau'_i \in \mathrm{L}(\tau', 1) : yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau'_i \in \mathrm{L}(\tau', 1)$: $yv_{i,1}$ is an integer $\in \{0, 1\}$
I5.	$\forall \tau'_i \in \mathrm{L}(\tau', 1)$: $yv_{i,2}$ is an integer $\in \{0, 1\}$
I6.	$\sum_{\tau'_i \in \mathrm{H1}(\tau',1) \cup \mathrm{L}(\tau',1) : u'_{i,1} > 1/2} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
I7.	$\sum_{\tau'_i \in \mathrm{H2}(\tau',1) \cup \mathrm{L}(\tau',1) : u'_{i,2} > 1/2} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

We can observe that if a task $\tau'_i \in \mathrm{H1}(\tau', 1)$ then it follows that the corresponding task $\tau_i \in \mathrm{H1}(\tau, 2/3)$. Analogously for tasks in $\mathrm{H2}(\tau, 2/3)$, $\mathrm{H12}(\tau, 2/3)$ and $\mathrm{L}(\tau, 2/3)$. Also, doing the following substitution: $u'_{i,1} = u_{i,1} \times \frac{3}{2}$ and $u'_{i,2} = u_{i,2} \times \frac{3}{2}$ and then rewriting the objective function and the first two and the last two constraints gives us:
The value of the objective function for an optimal solution of the following optimization problem is ≤ 1:

Minimize $(\frac{2}{3} \times zv) \times \frac{3}{2}$ subject to the following constraints:

I1.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,1} \times yv_{i,1} + \sum_{\tau_i \in \mathrm{H1}(\tau,2/3)} u_{i,1} \leq \frac{2}{3} \times zv \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,2} \times yv_{i,2} + \sum_{\tau_i \in \mathrm{H2}(\tau,2/3)} u_{i,2} \leq \frac{2}{3} \times zv \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3) : yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3)$: $yv_{i,1}$ is an integer $\in \{0, 1\}$
I5.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3)$: $yv_{i,2}$ is an integer $\in \{0, 1\}$
I6.	$\sum_{\tau_i \in \mathrm{H1}(\tau,2/3) \cup \mathrm{L}(\tau,2/3) : u_{i,1} > 1/3} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
I7.	$\sum_{\tau_i \in \mathrm{H2}(\tau,2/3) \cup \mathrm{L}(\tau,2/3) : u_{i,2} > 1/3} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

Substituting $\frac{2}{3} \times zv$ by zt and since claiming $zt \times \frac{3}{2} \leq 1$ is equivalent to claiming $zt \leq \frac{2}{3}$, we obtain:
The value of the objective function for an optimal solution of the following optimization problem is $\leq 2/3$:

Minimize zt subject to the following constraints:

I1.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,1} \times yv_{i,1} + \sum_{\tau_i \in \mathrm{H1}(\tau,2/3)} u_{i,1} \leq zt \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,2} \times yv_{i,2} + \sum_{\tau_i \in \mathrm{H2}(\tau,2/3)} u_{i,2} \leq zt \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3) : yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3)$: $yv_{i,1}$ is an integer $\in \{0, 1\}$
I5.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3)$: $yv_{i,2}$ is an integer $\in \{0, 1\}$
I6.	$\sum_{\tau_i \in \mathrm{H1}(\tau,2/3) \cup \mathrm{L}(\tau,2/3) : u_{i,1} > 1/3} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
I7.	$\sum_{\tau_i \in \mathrm{H2}(\tau,2/3) \cup \mathrm{L}(\tau,2/3) : u_{i,2} > 1/3} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

Substituting zt by zv gives us:
The value of the objective function for an optimal solution of the following optimization problem is $\leq 2/3$:

Minimize zv subject to the following constraints:

I1.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,1} \times yv_{i,1} + \sum_{\tau_i \in \mathrm{H1}(\tau,2/3)} u_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
I2.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,2} \times yv_{i,2} + \sum_{\tau_i \in \mathrm{H2}(\tau,2/3)} u_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
I3.	$\forall \tau_i \in \mathrm{L}(\tau,2/3) : yv_{i,1} + yv_{i,2} = 1$
I4.	$\forall \tau_i \in \mathrm{L}(\tau,2/3)$: $yv_{i,1}$ is an integer $\in \{0,1\}$
I5.	$\forall \tau_i \in \mathrm{L}(\tau,2/3)$: $yv_{i,2}$ is an integer $\in \{0,1\}$
I6.	$\sum_{\tau_i \in \mathrm{H1}(\tau,2/3) \cup \mathrm{L}(\tau,2/3): u_{i,1} > 1/3} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
I7.	$\sum_{\tau_i \in \mathrm{H2}(\tau,2/3) \cup \mathrm{L}(\tau,2/3): u_{i,2} > 1/3} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

Note that the optimization problem above is an MILP. We can relax the constraint on integrality of $yv_{i,1}$ and $yv_{i,2}$. This gives us a non-decreasing feasible region and hence the value of the objective function at an optimal solution is non-increasing. This gives us:
The value of the objective function for an optimal solution of the following optimization problem is $\leq 2/3$:

Minimize zv subject to the following constraints:

C1.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,1} \times yv_{i,1} + \sum_{\tau_i \in \mathrm{H1}(\tau,2/3)} u_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
C2.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,2} \times yv_{i,2} + \sum_{\tau_i \in \mathrm{H2}(\tau,2/3)} u_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
C3.	$\forall \tau_i \in \mathrm{L}(\tau,2/3) : yv_{i,1} + yv_{i,2} = 1$
C4.	$\forall \tau_i \in \mathrm{L}(\tau,2/3)$: $yv_{i,1}$ is a real number in $[0,1]$
C5.	$\forall \tau_i \in \mathrm{L}(\tau,2/3)$: $yv_{i,2}$ is a real number in $[0,1]$
C6.	$\sum_{\tau_i \in \mathrm{H1}(\tau,2/3) \cup \mathrm{L}(\tau,2/3): u_{i,1} > 1/3} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
C7.	$\sum_{\tau_i \in \mathrm{H2}(\tau,2/3) \cup \mathrm{L}(\tau,2/3): u_{i,2} > 1/3} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

Because of $yv_{i,1} + yv_{i,2} = 1$, it follows that $yv_{i,1} \leq 1$ and $yv_{i,2} \leq 1$. Hence, it is unnecessary to state that $yv_{i,1}$ and $yv_{i,2}$ are real numbers in $[0,1]$. Therefore, removing them does not impact the feasible region and also does not impact the value of the objective function at an optimal solution. This gives us:
The value of the objective function for an optimal solution of the following optimization problem is $\leq 2/3$:

Minimize zv subject to the following constraints:

C1.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,1} \times yv_{i,1} + \sum_{\tau_i \in \mathrm{H1}(\tau,2/3)} u_{i,1} \leq zv \times \left\lvert P^1(\pi) \right\rvert$
C2.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} u_{i,2} \times yv_{i,2} + \sum_{\tau_i \in \mathrm{H2}(\tau,2/3)} u_{i,2} \leq zv \times \left\lvert P^2(\pi) \right\rvert$
C3.	$\forall \tau_i \in \mathrm{L}(\tau,2/3) : yv_{i,1} + yv_{i,2} = 1$
C4.	$\forall \tau_i \in \mathrm{L}(\tau,2/3)$: $yv_{i,1}$ is a real number ≥ 0
C5.	$\forall \tau_i \in \mathrm{L}(\tau,2/3)$: $yv_{i,2}$ is a real number ≥ 0
C6.	$\sum_{\tau_i \in \mathrm{H1}(\tau,2/3) \cup \mathrm{L}(\tau,2/3): u_{i,1} > 1/3} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
C7.	$\sum_{\tau_i \in \mathrm{H2}(\tau,2/3) \cup \mathrm{L}(\tau,2/3): u_{i,2} > 1/3} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$

Consider the following call to the $\mathrm{TLP}_{\mathrm{CUT}}$ function, i.e., $\mathrm{TLP}_{\mathrm{CUT}}\big(\mathrm{L}(\tau,2/3),\ \pi,\ \mathrm{H1}(\tau,2/3),\ \mathrm{H2}(\tau,2/3),\ \mathrm{aot}\big)$. This gives:
$\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau,2/3), \pi, \mathrm{H1}(\tau,2/3), \mathrm{H2}(\tau,2/3), \mathrm{aot})$ =

Minimize zv subject to the following constraints:

C1.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} yv_{i,1} \times u_{i,1} + \sum_{\tau_i \in \mathrm{H1}(\tau,2/3)} u_{i,1} \leq \left\lvert P^1(\pi) \right\rvert \times zv$
C2.	$\sum_{\tau_i \in \mathrm{L}(\tau,2/3)} yv_{i,2} \times u_{i,2} + \sum_{\tau_i \in \mathrm{H2}(\tau,2/3)} u_{i,2} \leq \left\lvert P^2(\pi) \right\rvert \times zv$
C3.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3) : yv_{i,1} + yv_{i,2} = 1$
C4.	$\sum_{\tau_i \in \mathrm{aot}(\mathrm{H1}(\tau,2/3) \cup \mathrm{L}(\tau,2/3))} yv_{i,1} \leq \left\lvert P^1(\pi) \right\rvert$
C5.	$\sum_{\tau_i \in \mathrm{aot}(\mathrm{H2}(\tau,2/3) \cup \mathrm{L}(\tau,2/3))} yv_{i,2} \leq \left\lvert P^2(\pi) \right\rvert$
C6.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3)$: $yv_{i,1}$ is a real number ≥ 0
C7.	$\forall \tau_i \in \mathrm{L}(\tau, 2/3)$: $yv_{i,2}$ is a real number ≥ 0

Note that the earlier optimization problem is same as $\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau), \pi, \mathrm{H1}(\tau), \mathrm{H2}(\tau), \mathrm{aot})$. This gives us:
The value of the objective function for an optimal solution of the following optimization problem is $\leq 2/3$:
$\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau, 2/3), \pi, \mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot})$.

This gives us: $Z_{\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau,2/3),\pi,\mathrm{H1}(\tau,2/3),\mathrm{H2}(\tau,2/3),\mathrm{aot})} \leq 2/3$.

Hence, we have proven that:

$$\mathrm{sched}(\mathrm{OPT}, \tau', \pi) \Rightarrow$$
$$Z_{\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau,2/3),\pi,\mathrm{H1}(\tau,2/3),\mathrm{H2}(\tau,2/3),\mathrm{aot})} \leq 2/3$$

This states the lemma. □

COROLLARY 5.2. *Consider a task set τ and a two-type platform π. Let τ' be defined as:*

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \ \wedge \ u'_{i,2} = u_{i,2} \times 3/2$$

It then holds that:

$$\mathrm{sched}(\mathrm{OPT}, \tau', \pi) \Rightarrow$$
$$\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau, 2/3), \pi, \mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot}) \textit{ is feasible}.$$

PROOF. This follows from Lemma 5.1. □

LEMMA 5.3. *For each input τ and π to Algorithm 1 it holds that: When line 9 has finished execution, if the optimization problem is feasible then it holds that there are at most three tasks in τ that are fractionally type-assigned.*

PROOF. This follows from a well-known result about vertex solutions in Linear Programming. For a detailed proof, see Appendix A.1. □

LEMMA 5.4. *Consider* $\mathrm{FF}_{\mathrm{hf}}$*(ts, pl, ps, t) and assume that* $|\mathrm{aot}(\mathrm{ts}, t)| \leq |ps|$*. If* $\sum_{\tau_i \in \mathrm{ts}} u_{i,t} \leq (2/3) \times |ps|$ *then it holds that the execution of* $\mathrm{FF}_{\mathrm{hf}}$ *returns* $\mathrm{at} = \mathrm{ts}$

PROOF. We prove the claim by contradiction. Suppose that the lemma was incorrect. Then there is a set of tasks (ts), a two-type platform (pl), a set of processors (ps) and a type-id (t) for which it holds that:

$$\sum_{\tau_i \in \mathrm{ts}} u_{i,t} \leq (2/3) \times |ps| \tag{8}$$

and

$$\mathrm{FF}_{\mathrm{hf}} \text{ returns a set 'at' that is a strict subset of 'ts'} \tag{9}$$

Let us explore two cases:
Case (i): $\mathrm{aot}(\mathrm{at}, t) \neq \mathrm{aot}(\mathrm{ts}, t)$**.** Considering the execution of lines 1-9 and our assumption that $|\mathrm{aot}(\mathrm{ts}, t)| \leq |ps|$, we can see that this cannot happen.
Case (ii): $\mathrm{aot}(\mathrm{at}, t) = \mathrm{aot}(\mathrm{ts}, t)$**.** If this case would have happened then there must have been a task $\tau_i \in \mathrm{ts} \setminus \mathrm{aot}(\mathrm{ts}, t)$ such that when executing line 14, it was the case that:

$$\forall p \in ps, \text{it holds that } \sum_{\tau_j \in \mathrm{at}} (x_{j,p} \times u_{j,t}) + u_{i,t} > 1 \tag{10}$$

Because of Case (ii), it holds that when this line executed, τ_i has $u_{i,t} \leq 1/3$. Applying it on Expression (10) yields:

$$\forall p \in ps, \text{it holds that } \sum_{\tau_j \in \mathrm{at}} (x_{j,p} \times u_{j,t}) + 1/3 > 1 \tag{11}$$

Rewriting Expression (11) and adding them yields:

$$\sum_{p \in ps} \sum_{\tau_j \in \mathrm{at}} (x_{j,p} \times u_{j,t}) > 2/3 \times |ps|$$

Further rewriting of the above expression yields:

$$\sum_{\tau_j \in \mathrm{at}} u_{j,t} \sum_{p \in ps} x_{j,p} > 2/3 \times |ps| \tag{12}$$

Observe that, for each task, $\tau_j \in \mathrm{at}$, it holds that there is exactly one $p \in ps$ such that $x_{j,p} = 1$. Hence, we have for each task, $\tau_j \in \mathrm{ts}$, it holds that: $\sum_{p \in ps} x_{j,p} = 1$. Applying this on Expression (12) yields:

$$\sum_{\tau_j \in \mathrm{at}} u_{j,t} > 2/3 \times |ps| \tag{13}$$

Combining Expression (9) and Expression (13) yields:

$$\sum_{\tau_j \in \mathrm{ts}} u_{j,t} > 2/3 \times |ps|$$

This contradicts Expression (8).

Therefore, regardless of which case is true, it holds that, we obtain a contradiction. Hence, the statement of the lemma is true. □

LEMMA 5.5. *There is no π and τ and τ' such that*

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \textit{ and } u'_{i,2} = u_{i,2} \times 3/2$$

and

$$\mathrm{sched}\left(\mathrm{OPT}, \tau', \mathrm{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right)\right)$$

and

$$\mathrm{LPC} \textit{ declares FAILURE for inputs } \tau \textit{ and } \pi$$

PROOF. If the lemma would be incorrect then it holds that there is a π and τ and τ' such that

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \text{ and } u'_{i,2} = u_{i,2} \times 3/2 \tag{14}$$

and

$$\text{sched}\left(\text{OPT}, \tau', \text{mp}\left(\left|P^1\left(\pi\right)\right| - 3, \left|P^2\left(\pi\right)\right|, 1, \pi\right)\right) \tag{15}$$

and

$$\text{LPC declares FAILURE for inputs } \tau \text{ and } \pi \tag{16}$$

Because of Expression (16), it must have been that one of the lines where the algorithm declares FAILURE has been executed. We will first make a general remark about a class of these failures and then explore each failure individually. For the case that a failure happened at line 27,30,33,36,39,42 (Cases (1)-(6) below), we can reason as follows:

LPC must have executed line 9; so, it must hold that f='feasible'. Hence, $\text{TLP}_{\text{CUT}}\left(\text{L}(\tau, 2/3),\ \pi',\ \text{H1}(\tau, 2/3),\ \text{H2}(\tau, 2/3),\ \text{aot}\right)$ has a feasible solution. (Recall that $\pi' = \pi \setminus \text{rp}$, where rp is a set of three type-1 processors of π.)

Since the optimization problem is feasible, let us discuss the value of its objective function. Recall that Lemma 5.1 states that: *Consider a task set τ and a two-type platform π. Let τ' be defined as:*

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \ \wedge \ u'_{i,2} = u_{i,2} \times 3/2$$

It then holds that:

$$\text{sched}(\text{OPT}, \tau', \pi) \Rightarrow Z_{\text{TLP}_{\text{CUT}}(\text{L}(\tau,2/3),\pi,\text{H1}(\tau,2/3),\text{H2}(\tau,2/3),\text{aot})} \leq 2/3$$

Applying Lemma 5.1 on a platform with three fewer processors of type-1 gives us: *Consider a task set τ and a two-type platform π. Let τ' be defined as:*

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \ \wedge \ u'_{i,2} = u_{i,2} \times 3/2$$

It then holds that:

$$\begin{aligned}&\text{sched}\left(\text{OPT}, \tau', \text{mp}\left(\left|P^1\left(\pi\right)\right| - 3, \left|P^2\left(\pi\right)\right|, 1, \pi\right)\right) \Rightarrow \\ &\quad Z_{\text{TLP}_{\text{CUT}}\left(\text{L}(\tau,2/3),\text{mp}\left(\left|P^1(\pi)\right|-3,\left|P^2(\pi)\right|,1,\pi\right),\text{H1}(\tau,2/3),\text{H2}(\tau,2/3),\text{aot}\right)} \leq 2/3\end{aligned}$$

We know that Expression (15) is true and since the left-hand side predicate of the above implication is Expression (15), it follows that the right-hand side predicate of the implication is true. This gives us:

$$Z_{\text{TLP}_{\text{CUT}}\left(\text{L}(\tau,2/3),\text{mp}\left(\left|P^1(\pi)\right|-3,\left|P^2(\pi)\right|,1,\pi\right),\text{H1}(\tau,2/3),\text{H2}(\tau,2/3),\text{aot}\right)} \leq 2/3$$

Hence, we have: $z \leq 2/3$.

Therefore, for the case of failure on any of the lines 27, 30, 33, 36, 39 and 42, we have:

$$\begin{aligned}&\text{If the algorithm declares failure on line 27, 30, 33, 36, 39, 42} \\ &\qquad\qquad \text{then it holds that: } z \leq 2/3\end{aligned} \tag{17}$$

Let us now explore the individual cases:

Case (1): The algorithm declares failure on line 27. If this case would have happened then τ^{A2} is a strict subset of τ^2. Let us explore two cases:

Case (1a): $\sum_{\tau_i \in \tau^2} u_{i,2} > (2/3) \times \left|P^2\left(\pi\right)\right|$. Since we experienced Case (1), it holds that we have executed line 13 and evaluated its condition to true. Hence, we have:

$$z \leq 2/3 \tag{18}$$

Inspecting TLP_{CUT} and knowing that $z \leq 2/3$ gives us:

$$\sum_{\tau_i \in \text{L}(\tau,2/3) \cup \text{H2}(\tau,2/3)} y_{i,2} \times u_{i,2} \leq (2/3) \times \left|P^2\left(\pi\right)\right| \tag{19}$$

Let us partition $\mathrm{L}(\tau, 2/3)$ into $\mathrm{L1}$ and $\mathrm{L2}$ and applying it on Expression (19) gives us:

$$\sum_{\tau_i \in \mathrm{L1} \cup \mathrm{L2} \cup \mathrm{H2}(\tau,2/3)} y_{i,2} \times u_{i,2} \leq (2/3) \times \left|P^2(\pi)\right| \tag{20}$$

Recall that the definition of $\mathrm{L1}$ and $\mathrm{L2}$ it holds that:

$$\forall \tau_i \in \mathrm{L1} : y_{i,1} = 1 \tag{21}$$

$$\forall \tau_i \in \mathrm{L2} : y_{i,2} = 1 \tag{22}$$

Recall in $\mathrm{TLP}_{\mathrm{CUT}}$ we have a constraint $y_{i,1} + y_{i,2} = 1$ and clearly our values of Y satisfies that constraint. Applying this on Expression (21) gives us:

$$\forall \tau_i \in \mathrm{L1} : y_{i,2} = 0 \tag{23}$$

Using Expression (23) on Expression (20) gives us:

$$\sum_{\tau_i \in \mathrm{L2} \cup \mathrm{H2}(\tau,2/3)} y_{i,2} \times u_{i,2} \leq (2/3) \times \left|P^2(\pi)\right| \tag{24}$$

Because of Expression (22) and because of line 7 in our algorithm we obtain:

$$\sum_{\tau_i \in \mathrm{L2} \cup \mathrm{H2}(\tau,2/3)} u_{i,2} \leq (2/3) \times \left|P^2(\pi)\right| \tag{25}$$

Since $\tau^2 = \mathrm{H2}(\tau, 2/3) \cup \mathrm{L2}$, we get:

$$\sum_{\tau_i \in \tau^2} u_{i,2} \leq (2/3) \times \left|P^2(\pi)\right| \tag{26}$$

This contradicts the assumption of Case (1a).

Case (1b): $\sum_{\tau_i \in \tau^2} u_{i,2} \leq (2/3) \times \left|P^2(\pi)\right|$. From Lemma 5.4, we obtain that the bin-packing scheme in $\mathrm{FF}_{\mathrm{hf}}$ algorithm would succeed to assign all the tasks and then we would have $\tau^{\mathrm{A2}} = \tau^2$. This contradicts the Case (1).

Case (2): The algorithm declares failure on line 30. If this case would have happened then τ^{A1} is a strict subset of τ^1. The reasoning for this case is similar to the above case (replace $\left|P^2(\pi)\right|$ with $\left|P^1(\pi)\right| - 3$).

Case (3): The algorithm declares failure on line 33. If this case would have happened then $|\operatorname{aot}(\tau^2, 2)| > \left|P^2(\pi)\right|$. But then $\mathrm{TLP}_{\mathrm{CUT}}$ would be infeasible. And this contradicts Expression (17).

Case (4): The algorithm declares failure on line 36. If this case would have happened then $|\operatorname{aot}(\tau^1, 1)| > \left|P^1(\pi)\right| \setminus \mathrm{rp}$. But then $\mathrm{TLP}_{\mathrm{CUT}}$ would be infeasible. And this contradicts Expression (17).

Case (5): The algorithm declares failure on line 39. If this case would have happened then

$$\tau^{\mathrm{A}} \text{ is a strict subset of } \tau^{\mathrm{F}} \tag{27}$$

We know from Lemma 5.3 that in $\mathrm{TLP}_{\mathrm{CUT}}$ there are at most three fractionally type-assigned tasks from $\mathrm{L}(\tau, 2/3)$. And we know that the set rp has three processors of type-1. Hence, it is possible to assign each task in τ^{F} to a unique processor in rp. And indeed the execution of line 15, would therefore succeed and hence we would have:

$$\tau^{\mathrm{A}} = \tau^{\mathrm{F}} \tag{28}$$

This contradicts Expression (27).

Case (6): The algorithm declares failure on line 42. From this case we obtain $z > 2/3$. From Expression (17), we have, $z \leq 2/3$. This contradicts the case.

Case (7): The algorithm declares failure on line 45. If this case would have happened then $f \neq$ 'feasible' and hence the optimization problem

$$\mathrm{TLP}_{\mathrm{CUT}}(\mathrm{L}(\tau, 2/3), \pi', \mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot}) \text{ is infeasible} \tag{29}$$

Recall from Expression (15) that the following predicate holds true:

$$\mathrm{sched}\left(\mathrm{OPT}, \tau', \mathrm{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right)\right)$$

Applying this on Lemma 5.2 gives us:

$$\begin{aligned}\mathrm{sched}\left(\mathrm{OPT}, \tau', \mathrm{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right)\right) \Rightarrow \\ \mathrm{TLP}_{\mathrm{CUT}}\left(\mathrm{L}(\tau, 2/3), \mathrm{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right),\right. \\ \left.\mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot}\right) \text{ is feasible.}\end{aligned}$$

This gives us that:

$$\begin{aligned}\mathrm{TLP}_{\mathrm{CUT}}\left(\mathrm{L}(\tau, 2/3), \mathrm{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right),\right. \\ \left.\mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot}\right) \text{ is feasible.}\end{aligned}$$

Note that $\mathrm{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right)$ and π' have the same number of processors of each type and these processors are from π. Applying this on the above expression gives us:

$$\mathrm{TLP}_{\mathrm{CUT}}\left(\mathrm{L}(\tau, 2/3), \pi', \mathrm{H1}(\tau, 2/3), \mathrm{H2}(\tau, 2/3), \mathrm{aot}\right) \text{ is feasible.}$$

This contradicts Expression (29).

Case (8): The algorithm declares failure on line 48. If this case would have happened then there was a task in H12 and this would contradict Expression (15).

We see that all cases where LPC declares FAILURE lead to contradiction. Hence the lemma holds. □

LEMMA 5.6. *There is no π and τ such that*

$$\mathrm{sched}(\mathrm{OPT}, \tau, \pi)$$

and

$$\begin{aligned}&\mathrm{LPC} \textit{ declares FAILURE with inputs} \\ &\qquad \tau \textit{ and } \mathrm{mp}\left(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 3/2, \pi\right)\end{aligned}$$

PROOF. This follows from the previous lemma (obtained after a series of algebraic manipulations). For a detailed proof, see Appendix A.2. □

THEOREM 5.7.

$$\begin{aligned}&\mathrm{sched}\left(\mathrm{OPT}, \tau, \pi\right) \Rightarrow \\ &\mathrm{sched}\left(\mathrm{LPC}, \tau, \mathrm{mp}\left(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 3/2, \pi\right)\right)\end{aligned}$$

PROOF. Follows from Lemma 5.6 and the fact that when the algorithm declares success, each processor is utilized to at most 100% and each task is *integrally processor-assigned*. □

Note: The additional three processors in the platform that the algorithm, LPC, uses can either be of type-1 or type-2 or a combination of these two types. We have chosen all the additional processors to be of type-1, for ease of explanation. The result continues to hold for any combination of three additional processors as long as this information is input to the algorithm (so that it can form the remaining set of processors, rp, accordingly — Step 2 in Algorithm 1).

6. CONCLUSIONS AND DISCUSSION

The heterogeneous multiprocessor model is more generic than identical or uniform multiprocessor model, in terms of the systems that it can accommodate. Hence, it is interesting to study heterogeneous multiprocessor systems since a solution designed for such systems can also be applied to identical and uniform multiprocessor systems. In addition, two-type heterogeneous multiprocessor, which is a restricted version of heterogeneous model, is increasingly becoming relevant [AMD Inc. 2012; Apple Inc. 2012; Intel Corp. 2013b; 2013c; Nvidia Inc. 2013; Qualcomm Inc 2013; Samsung Inc. 2013; Texas Instruments 2012; Alben 2013; Intel Corp. 2013a]. Scheduling real-time tasks on two-type heterogeneous multiprocessors is a complex problem. In this work, we address this problem via a task assignment algorithm with a proven approximation ratio.

This work considered the problem of partitioning a given collection of implicit-deadline sporadic tasks upon a two-type heterogeneous multiprocessor platform. For this problem, a new algorithm, LPC, is proposed which provides the following guarantee: if a task set has a feasible partitioning on a two-type platform then, LPC succeeds in finding such a feasible partitioning as well but on a platform in which each processor is 1.5 times faster and has 3 additional processors. For systems with large number of processors, LPC has a better approximation ratio than the prior state-of-the-art algorithms. Also, to the best of our knowledge, this is the first work that utilizes cutting planes to attain an algorithm with provably good performance for assigning real-time tasks on heterogeneous multiprocessors.

ACKNOWLEDGMENTS

Copyright 2013 ACM This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center. This material has been approved for public release and unlimited distribution. DM-0000244.

REFERENCES

Jonah Alben. 2013. NVIDIA Brings Kepler, World's Most Advanced Graphics Architecture, to Mobile Devices. http://blogs.nvidia.com/blog/2013/07/24/kepler-to-mobile/. (2013).

AMD Inc. 2012. AMD Accelerated Processing Units. http://www.amd.com/fusion. (2012).

James Anderson and Anand Srinivasan. 2000. Early-release fair scheduling. In *Proceedings of the* 12^{th} *Euromicro conference on Real-time systems*. 35–43.

Björn Andersson and Konstantinos Bletsas. 2008. Sporadic Multiprocessor Scheduling with Few Preemptions. In *20th Euromicro Conference on Real-Time Systems*. 243–252.

Apple Inc. 2012. Apple A5X: Dual-core CPU and Quad-core GPU. http://www.apple.com/ipad/specs/. (2012).

Sanjoy Baruah. 2004a. Partitioning Real-Time Tasks Among Heterogeneous Multiprocessors. In 33^{rd} *International Conference on Parallel Processing*. 467–474.

Sanjoy Baruah. 2004b. Task partitioning upon heterogeneous multiprocessor platforms. In 10^{th} *IEEE International Real-Time and Embedded Technology and Applications Symposium*. 536–543.

José Correa, Martin Skutella, and José Verschae. 2012. The Power of Preemption on Unrelated Machines and Applications to Scheduling Orders. *Mathematics of Operations Research* 37, 2 (2012), 379–398.

Michael Dertouzos. 1974. Control Robotics: The Procedural Control of Physical Processes. In *Proceedings of IFIP Congress*. 807–813.

Ellis Horowitz and Sartaj Sahni. 1976. Exact and Approximate Algorithms for Scheduling Nonidentical Processors. *Journal of the ACM* 23, 2 (April 1976), 317–327.

Intel Corp. 2013a. Bay Trail: Multicore SoC Family for Mobile Devices. http://www.intel.com/newsroom/kits/idf/2013_fall/pdfs/bay_trail_fact_sheet.pdf. (2013).

Intel Corp. 2013b. Intel Atom Processor. http://ark.intel.com/products/family/29035. (2013).

Intel Corp. 2013c. The 4th Generation Intel Core i7 Processors. http://ark.intel.com/products/family/75023. (2013).

Klaus Jansen and Lorant Porkolab. 1999. Improved approximation schemes for scheduling unrelated parallel machines. In *Proceedings of the* 31^{st} *annual ACM symposium on Theory of computing*. 408–417.

Narendra Karmakar. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* 4, 4 (1984), 373–395.

Bernhard Korte and Jens Vygen. 2006. *Combinatorial Optimization: Theory and Algorithms* (3rd ed.). Springer.

Jan Lenstra, David Shmoys, and Éva Tardos. 1990. Approximation algorithms for scheduling unrelated parallel machines. *Math. Program.* 46 (1990), 259–271. Issue 3.

Greg Levin, Shelby Funk, Caitlin Sadowskin, Ian Pye, and Scott Brandt. 2010. DP-FAIR: A simple model for understanding optimal multiprocessor scheduling. In 22^{nd} *Euromicro Conference on Real-Time Systems*. 3–13.

Chang L. Liu and James W. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *Journal of the ACM* 20 (1973), 46–61.

Nvidia Inc. 2013. Tegra 4 Super Processors. http://www.nvidia.com/object/tegra.html. (2013).

Cynthia A. Phillips, Cliff Stein, Eric Torng, and Joel Wein. 1997. Optimal time-critical scheduling via resource augmentation. In *29th ACM Symposium on Theory of Computing*.

Chris N. Potts. 1985. Analysis of a linear programming heuristic for scheduling unrelated parallel machines. *Discrete Applied Mathematics* 10 (1985), 155–164.

Qualcomm Inc. 2013. Quad-core for next generation devices. http://www.qualcomm.com/snapdragon/specs. (2013).

Gurulingesh Raravi, Björn Andersson, and Konstantinos Bletsas. 2013. Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors. *Real-Time Systems* 49, 1 (2013), 29–72.

Gurulingesh Raravi, Björn Andersson, Konstantinos Bletsas, and Vincent Nélis. 2012. Task Assignment Algorithms for Two-Type Heterogeneous Multiprocessors. In 24^{th} *Euromicro Conference on Real-Time Systems*. 34–43.

Gurulingesh Raravi and Vincent Nélis. 2012. A PTAS for assigning sporadic tasks on two-type heterogeneous multiprocessors. In *Proceedings of the 33rd IEEE Real-Time Systems Symposium*. 117–126.

Samsung Inc. 2013. Samsung Exynos Processor. www.samsung.com/exynos/. (2013).

Texas Instruments. 2012. OMAP Mobile Processors. http://www.ti.com/omap. (2012).

Andreas Wiese, Vincenzo Bonifaci, and Sanjoy Baruah. 2013. Partitioned EDF scheduling on a few types of unrelated multiprocessors. *Real-Time Systems* 49, 2 (2013), 219–238.

A. SUPPORTING LEMMAS AND PROOFS

A.1. Proof of Lemma 5.3

Here, we restate Lemma 5.3 from Section 5 (see page 15) and prove the same.

LEMMA A.1. *For each input τ and π to Algorithm 1 it holds that: when line 9 has finished execution, if the optimization problem is feasible then it holds that there are at most three tasks in τ that are fractionally type-assigned.*

PROOF. Suppose that the claim is false. Then it holds that there is a τ and π such that if they are input to the function `solve`($\mathrm{TLP_{CUT}}$) then `solve`($\mathrm{TLP_{CUT}}$) outputs a solution in which four or more tasks are fractionally type-assigned. Then it must have been that in the solution output by `solve`($\mathrm{TLP_{CUT}}$), there were four or more tasks τ_i for which it holds that $0 < yv_{i,1} < 1$ and $0 < yv_{i,2} < 1$. Considering $\mathrm{TLP_{CUT}}$, we can observe that it has $2|\,\mathrm{L}\,| + 1$ variables and $2\,\mathrm{L}$ non-negativity constraints and $|\,\mathrm{L}\,| + 4$ other constraints. Hence, in the vertex solution, there are at most $|\,\mathrm{L}\,| + 4$ non-zero variables [Baruah 2004b]. Let us explore two cases:
Case 1: $zv = 0$**.** If this is the case then the vertex optimal solution produced by `solve`($\mathrm{TLP_{CUT}}$) has all type-integral assignments and hence this contradicts the claim that there were four of more fractionally type-assigned tasks.
Case 2: $zv > 0$**.** Since $zv > 0$, it follows that there are at most $|\,\mathrm{L}\,| + 3$ non-zero $yv_{i,t}$ values. Let Q denote the number of tasks that are fractionally type-assigned. From our assumption that the lemma is false, it follows that $Q \geq 4$. The number of non-zero values of Y is exactly $Q \times 2 + (|\,\mathrm{L}\,| - Q)$ because each fractionally type-assigned task provides us with two non-zero variables in Y and each integrally type-assigned task provides us with one non-zero variable in Y. Hence, we have:

$$Q \geq 4 \tag{30}$$

and

$$Q \times 2 + (\mathrm{L} - Q) \leq \mathrm{L} + 3 \tag{31}$$

Rewriting Expression (31) gives us:

$$Q \leq 3 \tag{32}$$

Expression (32) contradicts Expression (30). Thus it is impossible for the claim of the lemma to be false and hence the lemma holds. □

A.2. Proof of Lemma 5.6

Here, we restate Lemma 5.6 from Section 5 (see page 19) and prove the same.

LEMMA A.2. *There is no π and τ such that*

$$\mathrm{sched}(\mathrm{OPT}, \tau, \pi)$$

and

$$\mathrm{LPC} \textit{ declares FAILURE with inputs } \tau \textit{ and } \mathrm{mp}\left(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 3/2, \pi\right)$$

PROOF. Recall that Lemma 5.5 states that:
"There is no π and τ and τ' such that

$$\forall \tau'_i \in \tau' : u'_{i,1} = u_{i,1} \times 3/2 \text{ and } u'_{i,2} = u_{i,2} \times 3/2$$

and

$$\text{sched}\left(\text{OPT}, \tau', \text{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right)\right)$$

and

$$\text{the algorithm LPC declares FAILURE for inputs } \tau \text{ and } \pi\text{"}$$

Rewriting this so that it makes a statement about the same task set rather than two different (but related) task sets gives us that:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \text{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 2/3, \pi\right)\right)$$

and

$$\text{the algorithm LPC declares FAILURE for inputs } \tau \text{ and } \pi\text{"}$$

Scaling the processor speeds of the two platforms that are compared gives us that:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \text{mp}\left(\left|P^1(\pi)\right| - 3, \left|P^2(\pi)\right|, 1, \pi\right)\right)$$

and

$$\text{the algorithm LPC declares FAILURE for inputs } \tau \text{ and } \text{mp}\left(\left|P^1(\pi)\right|, \left|P^2(\pi)\right|, 3/2, \pi\right)\text{"}$$

Consider the statements above with π being replaced by $\text{mp}\left(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi\right)$. This gives us:
"There is no π and τ such that

$$\text{sched}\big(\text{OPT}, \tau, \text{mp}(|P^1(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))| - 3, |P^2(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))|, 1, \text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))\big)$$

and

$$\text{the algorithm LPC declares FAILURE for inputs } \tau \text{ and } \text{mp}(|P^1(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))|, |P^2(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))|, 3/2, \text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))\text{"}$$

Note that the last parameter indicates the platform from which we get processors to form a new platform. Hence the actual number of processors in the platform of the last parameter does not matter. This gives us:
"There is no π and τ such that

$$\text{sched}\big(\text{OPT}, \tau, \text{mp}(|P^1(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))| - 3, |P^2(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))|, 1, \text{mp}(\left|P^1(\pi)\right|, \left|P^2(\pi)\right|, 1, \pi))\big)$$

and

$$\text{the algorithm LPC declares FAILURE for inputs } \tau \text{ and } \text{mp}(|P^1(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))|, |P^2(\text{mp}(\left|P^1(\pi)\right| + 3, \left|P^2(\pi)\right|, 1, \pi))|, 3/2, \text{mp}(\left|P^1(\pi)\right|, \left|P^2(\pi)\right|, 1, \pi))\text{"}$$

Observe that $\text{mp}\left(\left|P^1\left(\pi\right)\right|, \left|P^2\left(\pi\right)\right|, 1, \pi\right) = \pi$. Applying that on the last parameter yields:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \text{mp}(|P^1(\text{mp}(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi))| - 3, |P^2(\text{mp}(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi))|, 1, \pi)\right)$$

and

the algorithm LPC declares FAILURE for inputs τ and

$$\text{mp}(|P^1(\text{mp}(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi))|, |P^2(\text{mp}(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi))|, 3/2, \pi)$$

"

Observe that $\left|P^2\left(\text{mp}\left(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi\right)\right)\right| = \left|P^2\left(\pi\right)\right|$. Applying that yields:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \text{mp}(|P^1(\text{mp}(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi))| - 3, \left|P^2\left(\pi\right)\right|, 1, \pi)\right)$$

and

the algorithm LPC declares FAILURE for inputs τ and

$$\text{mp}(|P^1(\text{mp}(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi))|, \left|P^2\left(\pi\right)\right|, 3/2, \pi)$$

"

Analogous observation holds for type-1 processors, i.e., $\left|P^1\left(\text{mp}\left(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 1, \pi\right)\right)\right| = \left|P^1\left(\pi\right)\right| + 3$. Applying this yields:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \text{mp}\left(\left|P^1\left(\pi\right)\right| + 3 - 3, \left|P^2\left(\pi\right)\right|, 1, \pi\right)\right)$$

and

the algorithm LPC declares FAILURE for inputs τ and

$$\text{mp}\left(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 3/2, \pi\right)$$

"

Clearly, $\left|P^1\left(\pi\right)\right| + 3 - 3 = \left|P^1\left(\pi\right)\right|$. Using it yields:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \text{mp}\left(\left|P^1\left(\pi\right)\right|, \left|P^2\left(\pi\right)\right|, 1, \pi\right)\right)$$

and

the algorithm LPC declares FAILURE for inputs τ and

$$\text{mp}\left(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 3/2, \pi\right)$$

"

Observe that $\text{mp}\left(\left|P^1\left(\pi\right)\right|, \left|P^2\left(\pi\right)\right|, 1, \pi\right) = \pi$. Using it yields:
"There is no π and τ such that

$$\text{sched}\left(\text{OPT}, \tau, \pi\right)$$

and

the algorithm LPC declares FAILURE for inputs τ and

$$\text{mp}\left(\left|P^1\left(\pi\right)\right| + 3, \left|P^2\left(\pi\right)\right|, 3/2, \pi\right)$$

"

This states the lemma. □