Minimum and Maximum Utilization Bounds for Multiprocessor Rate Monotonic Scheduling

José M. López, José L. Díaz, and Daniel F. García

Departamento de Informática, Universidad de Oviedo, Gijón 33204, Spain

December 4, 2003

Abstract

The utilization bound for real-time Rate Monotonic (RM) scheduling on uniprocessors is extended to multiprocessors with partitioning based scheduling. This allows fast schedulability tests to be performed on multiprocessors and quantifies the influence of key parameters, such as the number of processors and task sizes on the schedulability of the system. The multiprocessor utilization bound is a function of the allocation algorithm, so among all the allocation algorithms there exists at least one allocation algorithm providing the minimum multiprocessor utilization bound, and one allocation algorithm providing the maximum multiprocessor utilization bound. We prove that the multiprocessor utilization bound associated with the allocation heuristic *Worst Fit* (WF) coincides with that minimum if we use Liu & Layland's bound (LLB) as the uniprocessor schedulability condition. In addition, we present a class of allocation algorithms sharing the same multiprocessor utilization bound which coincides with the aforementioned maximum using LLB. The heuristics *First Fit Decreasing* (FFD) and *Best Fit Decreasing* (BFD) belong to this class. Thus, not even an optimal allocation algorithm can guarantee a higher multiprocessor utilization bound than that of FFD and BFD using LLB.

Finally, the pessimism of the multiprocessor utilization bounds is estimated through extensive simulations.

Keywords: Real-Time systems, multiprocessors, Rate Monotonic scheduling, allocation, utilization bounds.

1 Introduction

Multiprocessor scheduling is a challenging problem in the real-time systems theory. There are two main strategies when dealing with this problem: partitioning strategies and global strategies [1]. In a partitioning strategy, once a task is allocated to a processor, all of its instances are executed exclusively on that processor. In a global strategy, any instance of a task can be executed on any processor, or even be pre-empted and moved to a different processor, before it is completed.

Theoretically, global strategies provide higher schedulability than partitioning strategies. However, partitioning strategies have several advantages over global strategies. Firstly, the scheduling overhead associated with partitioning strategies is lower than that of global strategies. Secondly, partitioning strategies allow well known uniprocessor scheduling algorithms to be applied on each processor. Furthermore, Rate Monotonic (RM) and Earliest Deadline First (EDF) scheduling, which are optimal real-time uniprocessor scheduling algorithms [2], perform poorly when extended to global multiprocessor scheduling. The utilization bounds associated with global RM or EDF multiprocessor scheduling are not higher than one for any number of processors [3]. Nevertheless, these bounds can be greatly improved by placing restrictions on the tasks sizes, or by using some variations of global RM or EDF scheduling algorithms [4, 5].

In this article, we use the partitioning strategy, i.e, any task must always be executed on the same processor. Tasks are pre-emptively scheduled on each processor according to the RM algorithm. RM is a static priority scheduling algorithm that assigns each task a priority inversely proportional to its period, i.e, the smaller the period, the higher the priority. Thus, the allocation algorithm is the only degree of freedom in the system.

Finding optimal allocation algorithms is not practical, as the problem is NP-hard in the strong sense [6]. Several allocation algorithms have been proposed in the literature: simple allocation heuristics [3, 6, 7] and complex allocation algorithms such as those based on branch-and-bound [8] and simulated annealing techniques [9]. In this article, we focus on simple allocation heuristics.

Two different approaches are followed in the literature to establish the schedulability associated with a given allocation algorithm: simulation approaches and theoretical approaches.

In the simulation approach, task sets are randomly generated. Next, the average number of processors required to allocate task sets of a given total utilization is obtained. Uniprocessor exact tests [10], or uniprocessor sufficient tests [11] are commonly used to decide whether a given group of tasks fits into one processor. Nevertheless, simulation results should be considered carefully, since randomly generated task sets may not be representative of those that appear in practice.

The traditional theoretical approach focuses on the calculation of bounds for the metric $(N_{\mathrm{AA}}/N_{\mathrm{OPT}})$, for (*uniprocessor scheduling algorithm, allocation algorithm*) pairs [1, 3, 6, 7, 12, 13]. This metric gives the relationship between the number of processors required to schedule a task set using a given allocation algorithm AA, and the number of processors required using an optimal allocation algorithm OPT. This metric is useful in order to compare different allocation algorithms, but not to perform schedulability tests. There are several reasons for this. Firstly, N_{OPT} can not be calculated in polynomial time. Secondly, even if N_{OPT} were known, the utilization bound derived from the metric would be too pessimistic [14]. For example, for First Fit (FF) allocation $(N_{\mathrm{FF}}/N_{\mathrm{OPT}}) = 2.33$, as proved in [1]. A task set made up of 20 tasks, each with a utilization factor $(0.04+\varepsilon)$, may require two processors to be schedulable even using an optimal allocation algorithm, since LLB for 20 tasks is 0.7, less than the total utilization, $20(0.04+\varepsilon)$. Using the relation $(N_{\mathrm{FF}}/N_{\mathrm{OPT}}) = 2.33$, five processors would be necessary to guarantee the schedulability of the task set using the FF allocation. This gives a utilization bound not higher than $0.8/5 \approx 0.16$ per processor, too low to be useful. Oh and Son [1] refined the relation $(N_{\mathrm{FF}}/N_{\mathrm{OPT}})$ by considering the task sizes. Even so, the schedulability results obtained from these relations are too pessimistic when compared to the tight utilization bounds.

A new theoretical approach consists of calculating the utilization bounds associated with (*scheduling algorithm, allocation algorithm*) pairs, analogous to those known for uniprocessors. This approach has several interesting features: it allows us to carry out fast schedulability tests, and to quantify the influence of certain parameters, such as the number of processors, on schedulability. The major disadvantage of this approach is the sufficient but not necessary character of the associated schedulability tests. This approach was followed in [14] to obtain a lower limit, given by (1), on the utilization bound $U_{\mathrm{wc}}^{\text{RM-FF}}$ for multiprocessor RM scheduling with *First Fit allocation* (FF).

$$U_{\text{wc}}^{\text{RM-FF}}(n) \geq n(2^{1/2} - 1) \tag{1}$$

where n is the number of processors. From a practical point of view, Equation (1) states that any task set of total utilization less than $0.414n$ is schedulable in a multiprocessor made up of n processors using FF allocation and RM scheduling on each processor. The reader should compare the 0.414 utilization bound per processor with the utilization bound 0.16 per processor, obtained from relation $(N_{\text{FF}}/N_{\text{OPT}})$.

More recently, a tighter utilization bound for RM scheduling and FF allocation was presented in [15]. This bound considers not only the number of processors, but also the number of tasks and their sizes. The performance of the allocation algorithms depends largely on the task sizes. Thus, it is usual to place some kind of limit on the task sizes, e.g a maximum utilization factor, to improve the theoretical results [1].

The theoretical approach based on calculating utilizations bounds has also been developed recently for global multiprocessor RM scheduling. Under global RM scheduling, Andersson et al. [4] proved that the utilization bound is $n^2/(3n-2)$, when the utilization factor of any task is not higher than $n/(3n-2)$. They also present a variation of the global RM scheduling algorithm with the same utilization bound, $n^2/(3n-2)$, but without the previous restriction on task sizes. In addition, Baruah and Goossens [16] presented a generalization of the utilization bound for global RM scheduling by considering the case of uniform multiprocessors, i.e, they consider the possibility of multiprocessors made up of non-identical processors.

Our work makes the following theoretical contributions to the real-time multiprocessor schedulability analysis:

- The minimum utilization bound based on Liu & Layland's bound (LLB) evaluated among all the *reasonable allocation algorithms* is found. Reasonable allocation algorithms are those which fail to allocate a task only when there is no processor in the system with sufficient free capacity to hold the task [17]. Using LLB, a reasonable allocation algorithm is one that fails to allocate a task only when there is no processor in the system able to hold the task without violating LLB. The idea of restricting the study to reasonable allocation algorithms is to exclude theoretically possible, but impractical allocation algorithms, which would only complicate the mathematical description of the problem. In particular, the allocation heuristic *Worst First* (WF) provides this minimum utilization bound using LLB.

- A class of reasonable allocation algorithms, termed *Reasonable Allocation Decreasing* (RAD), is defined. These algorithms are proved to provide the maximum utilization bound using LLB among all the allocation algorithms (reasonable or not) for multiprocessor RM scheduling. The simple heuristics *First Fit Decreasing* (FFD) and *Best Fit Decreasing (BFD)*, described in [6], belong to this class. Thus, not even an optimal allocation algorithm can provide a higher multiprocessor utilization bound than that of FFD and BFD using LLB.

The rest of the article is organized as follows. Section 2 defines the computational system used. The minimum and maximum utilization bounds for multiprocessor RM scheduling using LLB are provided in Section 3. Section 4 presents the *Worst Fit* allocation heuristic (WF) and calculates its utilization bound, which coincides with the minimum using LLB. Section 5 proves the expression of the utilization bound for RAD allocation, which coincides with the maximum using LLB. The mathematical expressions of the minimum and maximum utilization bounds are analyzed in Section 6. Section 7 analyzes the pessimism of the utilization bounds. Finally, Section 8 presents our conclusions.

2 System definition

A task set consists of m independent periodic tasks $\{\tau_1, \ldots, \tau_m\}$, of computation times $\{C_1, \ldots, C_m\}$, periods $\{T_1, \ldots, T_m\}$, and hard deadlines equal to the task periods. The utilization factor u_i of any task τ_i, defined as $u_i = C_i/T_i$, is assumed to be $0 < u_i \leq \alpha \leq 1$, where α is the maximum reachable utilization factor for any task. Thus, α is a parameter of the task set which takes the “task sizes”into account. The total utilization of the task set, denoted by U, is the sum of the utilization factors of the tasks of which it is composed.

Tasks are allocated to an array of n identical processors $\{P_1, \ldots, P_n\}$. Once a task is allocated to a processor it is executed exclusively on that processor. Within each processor, tasks are pre-emptively scheduled using fixed priorities assigned according to the RM criterion [2].

Allocation is carried out using reasonable allocation (RA) algorithms [17]. A reasonable allocation algorithm is one which fails to allocate a task only when there is no processor in the system which can hold the task.

Whether a task fits into a processor depends on the uniprocessor scheduling algorithm, the uniprocessor schedulability condition and the tasks previously allocated to the processor. In this article, we use Liu & Layland's utilization bound (LLB) for uniprocessor RM scheduling [2] as the uniprocessor schedulability condition, which is given by (2)

$$U = \sum_{i=1}^{m} u_i \le m(2^{1/m} - 1) \tag{2}$$

Equation (2) states that any set of m tasks of total utilization $m(2^{1/m} - 1)$ or less is schedulable using RM scheduling on a uniprocessor. Thus, a task of utilization factor u_i fits into processor P_j, which already has m_j tasks allocated to it with total utilization U_j, if the $(m_j + 1)$ are schedulable, i.e, if $(m_j + 1)(2^{1/(m_j+1)} - 1) - U_j \ge u_i$.

Using LLB, a reasonable allocation algorithm is one which fails to allocate a task of utilization factor u_i to a multiprocessor made up of n processors, only when the task does not fit into any processor, i.e,

$$(m_j + 1)(2^{1/(m_j+1)} - 1) - U_j < u_i \qquad \text{for all } j = 1, \ldots, n \tag{3}$$

LLB is calculated by considering the worst combination of task periods and utilization factors [2], so it is only a sufficient schedulability condition for RM scheduling. In particular, LLB is derived from a worst-case task set in which all the tasks in the processor have the same utilization factors and the periods fulfill the relation $T_{k+1}/T_k = 2^{1/m_j}$, i.e, $T_2 = 2^{1/m_j} T_1$, $T_3 = 2^{1/m_j} T_2$, and so on. Despite this, in this article we assume that a task set fits into one processor if, and only if, LLB is fulfilled. Therefore, the multiprocessor utilization bounds provided in the article are valid, but may not be tight, i.e, it may be possible to find higher multiprocessor utilization bounds that still guarantee the schedulability under multiprocessor RM scheduling. The multiprocessor utilization bounds could be made tight using necessary and sufficient schedulability conditions for uniprocessor RM scheduling [18] in the theorems, but they are too complex.

At this point, the notation should be clarified. The multiprocessor utilization bound for an allocation algorithm AA using LLB is denoted by $U_{\text{wc}}^{\text{LLB-AA}}$, while the tight multiprocessor utilization bound is denoted by $U_{\text{wc}}^{\text{RM-AA}}$. In general, $U_{\text{wc}}^{\text{RM-AA}} \geq U_{\text{wc}}^{\text{LLB-AA}}$, so $U_{\text{wc}}^{\text{LLB-AA}}$ may be pessimistic.

3 Minimum and maximum utilization bounds

The multiprocessor utilization bound $U_{\text{wc}}^{\text{LLB-RA}}$, associated with any reasonable allocation algorithm, RA, and LLB is in the interval $[L_{\text{LLB}}, H_{\text{LLB}}]$. This interval is defined as follows:

$$L_{\text{LLB}} = \min_{RA} U_{\text{wc}}^{\text{LLB-RA}} ; \qquad H_{\text{LLB}} = \max_{RA} U_{\text{wc}}^{\text{LLB-RA}}$$

The calculation of this interval gives the worst and best utilization bounds that can be expected from all the reasonable allocation algorithms beforehand.

Before calculating the expressions of L_{LLB} and H_{LLB}, it is necessary to introduce the parameter β_{LLB}. Parameter β_{LLB} is the maximum number of tasks of utilization factor α which fit into one processor using LLB for RM scheduling. β_{LLB} can be expressed as a function of α.

Lemma 1. *[15]*

$$\beta_{LLB} = \left\lfloor \frac{1}{\log_2(\alpha+1)} \right\rfloor \tag{4}$$

Proof. From the definition of β_{LLB}, β_{LLB} tasks of utilization factor α fit into one processor using LLB. Applying LLB this means that $\beta_{\text{LLB}}\alpha \leq \beta_{\text{LLB}}(2^{1/\beta_{\text{LLB}}} - 1)$. Finding β_{LLB} we obtain $\beta_{\text{LLB}} \leq 1/\log_2(\alpha+1)$. Since β_{LLB} is a natural number we get

$$\beta_{\text{LLB}} \leq \left\lfloor \frac{1}{\log_2(\alpha+1)} \right\rfloor \tag{5}$$

Because β_{LLB} is the maximum number of tasks of utilization factor α that fit into one processor using LLB, $(\beta_{\text{LLB}}+1)$ tasks of utilization factor α do not fit into one processor without violating LLB. Thus, $(\beta_{\text{LLB}}+1)\alpha > (\beta_{\text{LLB}}+1)(2^{1/(\beta_{\text{LLB}}+1)} - 1)$. Finding β_{LLB} we obtain $\beta_{\text{LLB}} > 1/\log_2(\alpha+1) - 1$. Since β_{LLB} is a natural number we get

$$\beta_{\text{LLB}} \geq \left\lfloor \frac{1}{\log_2(\alpha+1)} \right\rfloor \tag{6}$$

The lemma is proved from (5) and (6). □

Any multiprocessor made up of *n* processors can allocate at least $n\beta_{\text{LLB}}$ tasks of arbitrary utilization factors (less than or equal to α). Thus, any task set fulfilling $m \leq n\beta_{\text{LLB}}$ is trivially schedulable using RM scheduling together with any reasonable allocation algorithm. Henceforth, we will assume $m > n\beta_{\text{LLB}}$, as otherwise there would be no point in obtaining the utilization bounds.

Theorem 1 will provide a lower limit on the multiprocessor utilization bound associated with any reasonable allocation algorithm and LLB. Section 4 will present an upper limit on the utilization bound for one reasonable allocation algorithm, the WF heuristic, which coincides with the previous lower limit. Therefore, L_{LLB} and the utilization bound for WF allocation must coincide, and be equal to both limits.

Theorem 2 will provide an upper limit on the utilization bound associated with any allocation algorithm that is based on LLB, reasonable or not. Section 5 will present a lower limit on the utilization bound associated with some reasonable allocation algorithms, the heuristics in the class RAD, which coincides with the previous upper limit. Therefore, H_{LLB} and the utilization bound for the class RAD must coincide, and be equal to both limits. Furthermore, since the upper limit given by Theorem 2 applies to any allocation algorithm, reasonable or not, H_{LLB} is the maximum utilization bound among all the allocation algorithms using LLB.

Next, Theorem 1 provides a lower limit on the utilization bound for any reasonable allocation algorithm. This utilization bound will be denoted by $U_{\text{wc}}^{\text{LLB-RA}}(m,n,\alpha)$. At most, it depends on all the system parameters, i.e, the number of tasks, *m*, the number of processors, *n*, and the maximum reachable utilization factor, α.

Theorem 1. *Let RA be any reasonable allocation algorithm. If $m > n\beta_{LLB}$, it follows that $U_{wc}^{LLB\text{-}RA}(m,n,\alpha) \geq n_a U_a + n_b U_b - (n-1)\alpha$, where*

$$n_a = m+n-1-\left\lfloor \frac{m+n-1}{n} \right\rfloor n \qquad n_b = n - n_a$$
$$U_a = \left\lceil \frac{m+n-1}{n} \right\rceil \left(2^{1/\lceil \frac{m+n-1}{n} \rceil} - 1\right) \qquad U_b = \left\lfloor \frac{m+n-1}{n} \right\rfloor \left(2^{1/\lfloor \frac{m+n-1}{n} \rfloor} - 1\right)$$

Proof. Let $\{\tau_1, \ldots, \tau_m\}$ be a set of m tasks which does not fit into the multiprocessor using LLB on each processor. There are tasks of the set which are allocated to processors, and tasks which are not allocated. Let us change the indices in the set so that the tasks which were not allocated have the last indices in the set. Let τ_k be the first task in the set which was not allocated to any processor, after the change of indices. Since the allocation algorithm is reasonable, from (3) we get

$$(m_j+1)(2^{1/(m_j+1)}-1) - U_j < u_k \qquad \text{for all } j = 1, \ldots, n \tag{7}$$

where U_j is the total utilization of the tasks previously allocated to processor P_j, m_j is the number of these tasks, and u_k is the utilization factor of task τ_k. The total utilization of the whole set, U, fulfils

$$U = \sum_{i=1}^{m} u_i \geq \sum_{i=1}^{k} u_i = \sum_{j=1}^{n} U_j + u_k \tag{8}$$

From (7) we get

$$\sum_{j=1}^{n} U_j > \sum_{j=1}^{n} \left((m_j+1)(2^{1/(m_j+1)}-1) - u_k\right) = \sum_{j=1}^{n} (m_j+1)(2^{1/(m_j+1)}-1) - nu_k$$

Substituting this inequality into (8)

$$U > \sum_{j=1}^{n} (m_j+1)(2^{1/(m_j+1)}-1) - (n-1)u_k$$

From the system definition, all the utilization factors are less than or equal to α, so $u_k \leq \alpha$ and

$$U > \sum_{j=1}^{n} (m_j+1)(2^{1/(m_j+1)}-1) - (n-1)\alpha$$

In order to simplify the above expression, let us define $I_j = (m_j+1)$ and $g(I_1, \ldots, I_n) =$

$\sum_{j=1}^{n} I_j(2^{1/I_j}-1)$. We can now write

$$U > g(I_1,\ldots,I_n) - (n-1)\alpha \tag{9}$$

τ_k is the first task which does not fit into the multiprocessor. Thus, one constraint of the m_j values is that $\sum_{j=1}^{n} m_j = (k-1)$. This constraint is equivalent to the constraint $\sum_{j=1}^{n} I_j = (k+n-1)$. Bearing this last constraint in mind and applying Proposition 1 of the Appendix for $M = (k+n-1)$

$$\begin{aligned} g(I_1,\ldots,I_n) \geq \left(k+n-1-\left\lfloor\frac{k+n-1}{n}\right\rfloor n\right)\left\lceil\frac{k+n-1}{n}\right\rceil\left(2^{\frac{1}{\left\lceil\frac{k+n-1}{n}\right\rceil}}-1\right)+ \\ \left(1-k+\left\lfloor\frac{k+n-1}{n}\right\rfloor n\right)\left\lfloor\frac{k+n-1}{n}\right\rfloor\left(2^{\frac{1}{\left\lfloor\frac{k+n-1}{n}\right\rfloor}}-1\right) \end{aligned} \tag{10}$$

The right-hand term in (10) decreases as k increases, as is indicated just after Proposition 1 in the Appendix. Index k is in the interval $[1,m]$, since τ_k is a task of the set of m tasks. Therefore, for $k=m$ we obtain the minimum of the right-hand term in (10). Hence, from equations (9) and (10), and considering the definitions of n_a, n_b, U_a and U_b, we get

$$U > n_a U_a + n_b U_b - (n-1)\alpha$$

Any task set which does not fit into the processors using LLB fulfils the previous expression. Consequently, any task set of total utilization less than or equal to $n_a U_a + n_b U_b - (n-1)\alpha$ fits into the processors using LLB, and $U_{\text{wc}}^{\text{LLB-RA}}(m,n,\alpha) \geq n_a U_a + n_b U_b - (n-1)\alpha$ $\square$

We will prove that $U_{\text{wc}}^{\text{LLB-WF}}(m,n,\alpha \leq \ln 2) \leq n_a U_a + n_b U_b - (n-1)\alpha$ in Section 4. Since WF is a reasonable allocation algorithm we can state that

$$L_{\text{LLB}}(m,n,\alpha \leq \ln 2) = n_a U_a + n_b U_b - (n-1)\alpha$$

Next, we provide an intuitive idea about what n_a, n_b, U_a and U_b represent. This will be useful in the proof of Theorem 3 in Section 4. After dividing $(m-1)$ tasks quasi-equitably among n processors, there are n_a processors with $\lceil (m-1)/n \rceil$ tasks, and $n_b = (n-n_a)$ processors with $\lfloor (m-1)/n \rfloor$ tasks, i.e, one less task. U_a is the uniprocessor utilization bound for each of the n_a processors after receiving one more task. Likewise, U_b is the uniprocessor utilization bound for each of the n_b processors after receiving one more task.

Theorem 2 provides an upper limit on the multiprocessor utilization bound with any allocation algorithm that is based on LLB, reasonable or not.

Theorem 2. *Let AA be an arbitrary allocation algorithm. If $m > n\beta_{LLB}$, it follows that $U_{wc}^{LLB\text{-}AA} \leq (n\beta_{LLB}+1)(2^{1/(\beta_{LLB}+1)}-1)$*

Proof. We will prove that a set of m tasks $\{\tau_1,\ldots,\tau_m\}$ exists, with utilization factors $0 < u_i \leq \alpha$ for all $i = 1,\ldots,m$, and total utilization $(n\beta_{\text{LLB}}+1)(2^{1/(\beta_{\text{LLB}}+1)}-1)+\varepsilon$, with $\varepsilon \to 0^+$, which does not fit into n processors using any allocation algorithm and LLB on each processor. The proof will be divided into four parts:

1. The task set is presented.
2. The task set is proved to be made up of m tasks and to have a total utilization equal to $(n\beta_{\text{LLB}}+1)(2^{1/(\beta_{\text{LLB}}+1)}-1)+\varepsilon$.
3. The utilization factors of the task set are proved to be valid, that is, $0 < u_i \leq \alpha$.
4. The claim that the task set does not fit into the multiprocessor is proved.

Part 1. The set of m tasks is composed of two subsets: a first subset with $(m-n\beta_{\text{LLB}}-1)$ tasks, and a second subset with $(n\beta_{\text{LLB}}+1)$ tasks. All the tasks of the first subset have the same utilization factor of value $u_i = \varepsilon/m$, where $i = 1,\ldots,(m-n\beta_{\text{LLB}}-1)$. All the tasks of the second subset have the same utilization factor of value $u_i = (2^{1/(\beta_{\text{LLB}}+1)}-1)+\frac{\varepsilon}{m}$, where $i = (m-n\beta_{\text{LLB}}),\ldots,m$.

Part 2. It is simple to check that the task set is composed of m tasks of total utilization $(n\beta_{\text{LLB}}+1)(2^{1/(\beta_{\text{LLB}}+1)}-1)+\varepsilon$.

Part 3. It is also necessary to prove that the utilization factors of both subsets are valid, i.e, $0 < u_i \leq \alpha$ for all $i = 1,\ldots,m$.

Check of the utilization factors of the first subset. By choosing an small value for ε, we obtain $0 < u_i = \frac{\varepsilon}{m} \leq \alpha$.

Check of the utilization factors of the second subset. By definition of β_{LLB}, $(\beta_{\text{LLB}}+1)$ tasks of utilization factor α do not fit into one processor, therefore $(\beta_{\text{LLB}}+1)\alpha > (\beta_{\text{LLB}}+1)(2^{1/(\beta_{\text{LLB}}+1)}-1)$, and

$$\alpha > (2^{1/(\beta_{\text{LLB}}+1)}-1) > 0 \tag{11}$$

It is always possible to find one real number between two real numbers. Hence, a positive value of ε exists such that $\alpha > (2^{1/(\beta_{\text{LLB}}+1)} - 1) + \frac{\varepsilon}{m} = u_i > 0$. This proves that the utilization factors of the second subset are less than α when $\varepsilon \to 0^+$, and is higher than zero.

Part 4. There are $(n\beta_{\text{LLB}} + 1)$ tasks in the second subset. Hence, at least one processor of the n available should allocate $(\beta_{\text{LLB}} + 1)$ or more of these tasks. However, no processor can allocate $(\beta_{\text{LLB}} + 1)$ or more tasks of the second subset, since $(\beta_{\text{LLB}} + 1)$ of these tasks together have a utilization over LLB.

$$(\beta_{\text{LLB}} + 1)\left((2^{1/(\beta_{\text{LLB}}+1)} - 1) + \frac{\varepsilon}{m}\right) > (\beta_{\text{LLB}} + 1)(2^{1/(\beta_{\text{LLB}}+1)} - 1)$$

We conclude that the proposed task set of total utilization $U = (n\beta_{\text{LLB}} + 1)(2^{1/(\beta_{\text{LLB}}+1)} - 1) + \varepsilon$ does not fit into n processors when $\varepsilon \to 0^+$ using LLB on each processor, so the utilization bound $U_{\text{wc}}^{\text{LLB-AA}}$ must be less than or equal to $(n\beta_{\text{LLB}} + 1)(2^{1/(\beta_{\text{LLB}}+1)} - 1)$.

Remark: the tasks of the first subset are necessary in the proof only to fulfill the restriction of having m tasks. $\square$

4 Utilization bound for Worst Fit allocation

This section shows that the allocation algorithm termed Worst Fit (WF) is the worst reasonable allocation algorithm in terms of the multiprocessor utilization bound using LLB on each processor.

The WF algorithm allocates each task to the processor with the highest remaining capacity of all the processors with sufficient capacity to hold the task. Tasks are allocated one by one following the sequence $\{\tau_1, \ldots, \tau_m\}$. If two or more processors have the same remaining capacity, we assume that the task is allocated to the processor with the lowest index among those with the lowest remaining capacity. Using LLB, the remaining capacity of processor P_j is given by the expression $(m_j + 1)(2^{1/(m_j+1)} - 1) - U_j$. For example, consider a task of utilization factor 0.2, which we try to allocate to a multiprocessor made up of two processors, τ_1 and τ_2, using the WF algorithm. Let us suppose that processor τ_1 already holds two tasks of total utilization 0.5, that is, $m_1 = 2$ and $U_1 = 0.5$. Let us also suppose that processor τ_2 already holds three tasks of total utilization 0.49, that is, $m_2 = 3$ and $U_2 = 0.49$. The remaining capacity of τ_1 is

$(3(2^{1/3}-1)-0.5) \approx 0.32$, while the remaining capacity of τ_2 is $(4(2^{1/4}-1)-0.49) \approx 0.27$. Thus, both processors may allocate the task. Processor τ_1 will be the one that receives the task, as it is the processor with the highest remaining capacity of those with enough remaining capacity to hold the task.

The WF allocation has no practical value. Other allocation algorithms, such us FF and FFD perform better and have similar or less complexity than WF. However, the WF allocation is interesting from a theoretical perspective. WF provides the lowest utilization bound we can expect from any allocation algorithm based on LLB. In addition, we will prove that the multiprocessor utilization bound for any reasonable allocation algorithm under dynamic allocation coincides with the utilization bound for (static) WF allocation, given by Corollary 1. However, as we will remark at the end of this section, this bound can be applied only to steady states.

Next, Theorem 3 gives an upper limit on the multiprocessor utilization bound for WF allocation and LLB, which is a function of α at least. This upper limit coincides with the lower limit provided by Theorem 1 for any reasonable allocation algorithm, and therefore with the utilization bound for WF allocation, given by Corollary 1. The result is restricted to $\alpha \leq \ln 2$ in order to simplify the proof. In addition, for $\alpha > \ln 2$ the upper limit is too low to be useful in practice, as indicated in Section 6.

The terms n_a, n_b, U_a and U_b in the statement of Theorem 3 are defined in Theorem 1. The reader should refer to the intuitive description of these parameters given after the proof of Theorem 1, in order to better understand the proof of Theorem 3.

Theorem 3. *If $m > n\beta_{LLB}$, it follows that $U_{wc}^{LLB\text{-}WF}(m,n,\alpha \leq \ln 2) \leq n_a U_a + n_b U_b - (n-1)\alpha$*

Proof. We will prove the existence of a set of m tasks, $\{\tau_1, \ldots, \tau_m\}$, of utilization factors less than or equal to α, and total utilization $n_a U_a + n_b U_b - (n-1)\alpha + \varepsilon$ with $\varepsilon \to 0^+$, which does not fit into the processors using the allocation algorithm WF and LLB on each processor. The proof will be divided into four parts:

1. The task set is presented.
2. The task set is proved to be made up of m tasks and to have total utilization equal to $n_a U_a + n_b U_b - (n-1)\alpha + \varepsilon$.
3. The utilization factors of the task set are proved to be valid, that is, $0 < u_i \leq \alpha$.
4. The claim that the task set does not fit into the multiprocessor is proved.

Part 1. The set of m tasks is built as follows, strictly in the order indicated. There are $\lfloor (m-1)/n \rfloor$ subsets of $n = (n_a + n_b)$ tasks each. All these subsets are made up of n_a tasks of utilization factor

$$u_a = \frac{U_a - \alpha}{\lceil (m-1)/n \rceil} + \frac{\varepsilon}{m-1} \tag{12}$$

followed by n_b tasks of utilization factor

$$u_b = \frac{U_b - \alpha}{\lfloor (m-1)/n \rfloor} + \frac{\varepsilon}{m-1} \tag{13}$$

Following the previous $\lfloor (m-1)/n \rfloor$ subsets, there are n_a tasks of utilization factor u_a. Finally, there is the last task, τ_m, of utilization factor α.

Part 2. As a whole, the task set is made up of $\left(\left\lfloor \frac{m-1}{n} \right\rfloor + 1\right) n_a$ tasks of utilization factor u_a, $\left(\left\lfloor \frac{m-1}{n} \right\rfloor\right) n_b$ tasks of utilization factor u_b and one task of utilization factor α. The number of tasks in the set is $\left\lfloor \frac{m-1}{n} \right\rfloor (n_a + n_b) + n_a + 1$. Substituting the values of n_a and n_b

$$\left\lfloor \frac{m-1}{n} \right\rfloor n + m + n - 1 - \left\lfloor \frac{m+n-1}{n} \right\rfloor n + 1$$

Bearing in mind that $\left\lfloor \frac{m+n-1}{n} \right\rfloor = \left\lfloor \frac{m-1}{n} \right\rfloor + 1$, it follows that the task set is made up of m tasks.

In order to check the correctness of the total utilization, we will consider two cases: $(m-1)$ is a multiple of n, and $(m-1)$ is not a multiple of n. In the first case, $n_a = 0$, $n_b = n$, and $u_b = \frac{U_b - \alpha}{\frac{m-1}{n}} + \frac{\varepsilon}{m-1}$. The total utilization is $(m-1)u_b + \alpha$, which coincides with $n_a U_a + n_b U_b - (n-1)\alpha + \varepsilon$. In the second case, $\left\lceil \frac{m-1}{n} \right\rceil = \left\lfloor \frac{m-1}{n} \right\rfloor + 1 = \left\lfloor \frac{m+n-1}{n} \right\rfloor$. The total utilization becomes

$$\left(\left\lfloor \frac{m-1}{n} \right\rfloor + 1\right) n_a \left(\frac{U_a - \alpha}{\left\lfloor \frac{m-1}{n} \right\rfloor + 1} + \frac{\varepsilon}{m-1} \right) + \left\lfloor \frac{m-1}{n} \right\rfloor n_b \left(\frac{U_b - \alpha}{\left\lfloor \frac{m-1}{n} \right\rfloor} + \frac{\varepsilon}{m-1} \right) + \alpha =$$
$$n_a(U_a - \alpha) + n_b(U_b - \alpha) + \alpha + \frac{\varepsilon}{m-1} \left(\left\lfloor \frac{m-1}{n} \right\rfloor (n_a + n_b) + n_a \right)$$

Taking the expressions of n_a and n_b into account we get the total utilization $n_a U_a + n_b U_b - (n-1)\alpha + \varepsilon$

Part 3. It is necessary to prove that the utilization factors of all the tasks are valid, i.e, $0 < u_i \le \alpha$ for $i = 1, \ldots, n$.

For $x > 0$, function $x(2^{1/x} - 1)$ decreases as x increases. Therefore, it follows that $U_b \ge$

$U_a > \lim_{x\to\infty} x(2^{1/x}-1) = \ln 2 \geq \alpha$, and so u_a, u_b and the utilization factor of all the tasks are higher than zero. The utilization factor of the last task is α, and therefore it is less than or equal to α. In addition, we must prove that $u_a \leq \alpha$ and $u_b \leq \alpha$ in order to prove that the proposed task set is valid. It is sufficient to prove that $u_b \leq \alpha$, since $U_a \leq U_b$ and so $u_a \leq u_b$.

Substituting the value of U_b in the definition of u_b

$$u_b = \frac{\left\lfloor \frac{m+n-1}{n} \right\rfloor \left(2^{1/\left\lfloor \frac{m+n-1}{n} \right\rfloor} - 1\right) - \alpha}{\lfloor (m-1)/n \rfloor} + \frac{\varepsilon}{m-1}$$

From the hypothesis of the Theorem $m > n\beta_{\text{LLB}}$. Since β_{LLB} is a natural number, $(m-1) \geq n\beta_{\text{LLB}}$, $\lfloor (m-1)/n \rfloor \geq \beta_{\text{LLB}} \geq 1$, and $\lfloor (m+n-1)/n \rfloor \geq (\beta_{\text{LLB}}+1)$. In addition, for $x > 0$, function $x(2^{1/x}-1)$ decreases as x increases. Hence,

$$\left\lfloor \frac{m+n-1}{n} \right\rfloor \left(2^{1/\left\lfloor \frac{m+n-1}{n} \right\rfloor} - 1\right) \leq (\beta_{\text{LLB}}+1)(2^{1/(\beta_{\text{LLB}}+1)} - 1)$$

From, inequality (11), presented in the previous section, $\alpha > (2^{1/(\beta_{\text{LLB}}+1)} - 1)$ and we obtain $u_b < (2^{1/(\beta_{\text{LLB}}+1)} - 1) + \frac{\varepsilon}{m-1}$. Thus, by making ε close to zero we finally get $u_b < \alpha$.

Part 4. Next, we will prove that the task set does not fit into the multiprocessor. The first $(m-1)$ tasks are allocated by the WF heuristic as indicated in Figure 1. Numbers within parenthesis in Figure 1 represent task indices. Each row represents the tasks allocated to one processor and horizontal distances indicate utilization.

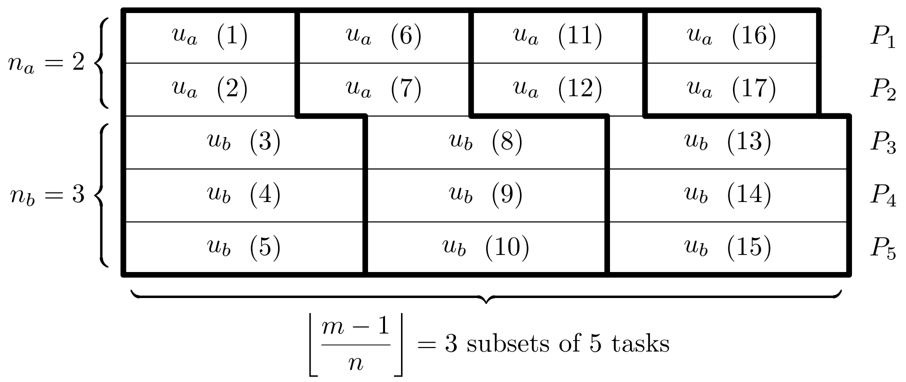

Figure 1: Example of allocation of the first $(m-1)$ tasks in Theorem 3, for $m = 18$ and $n = 5$.

As a result of the allocation of the first $(m-1)$ tasks, the first n_a processors hold $\lceil (m-1)/n \rceil$ tasks of utilization factors u_a. These processors may hold one additional task of utilization factor $U_a - \lceil (m-1)/n \rceil u_a = \alpha - \varepsilon \lceil (m-1)/n \rceil / (m-1)$. Therefore, the last task of utilization factor α does not fit into any of these processors using LLB. Nor can the remaining n_b processors hold the last task because at most they can hold one additional task of utilization factor $U_b - \lfloor (m-1)/n \rfloor u_b = \alpha - \varepsilon \lfloor (m-1)/n \rfloor / (m-1)$.

We conclude that the proposed task set of total utilization $n_aU_a + n_bU_b - (n-1)\alpha$ does not fit into n processors using LLB when $\varepsilon \to 0^+$, so the utilization bound $U_{\mathrm{wc}}^{\mathrm{LLB\text{-}WF}}(m,n,\alpha)$ must be less than or equal to $n_aU_a + n_bU_b - (n-1)\alpha$. □

Corollary 1 provides the multiprocessor utilization bound for WF allocation and LLB.

Corollary 1. *If $m > n\beta_{LLB}$, it follows that $U_{wc}^{LLB\text{-}WF}(m,n,\alpha \leq \ln 2) = n_aU_a + n_bU_b - (n-1)\alpha$*

Proof. : The proof is direct from Theorem 1 and Theorem 3. □

It can be proved that the multiprocessor utilization bound for any reasonable allocation algorithm under dynamic allocation coincides with the utilization bound for (static) WF allocation, given by Corollary 1. Dynamic allocation appears when the task set changes to deal with variable environments. In variable environments, some tasks are removed, increasing the available capacity of the processors, while other tasks are added to the system and allocated using the available processor capacities. There are also tasks that can not be stopped, which are common to different environments. These tasks can not migrate among processors, as this would require stopping and starting them again.

With respect to the allocation algorithm, there are tasks previously allocated to processors, which are common to different environments, and new tasks that must be allocated within the remaining room as the environment changes. Independently of the reasonable allocation algorithm, the worst-case situation appears when the previously allocated tasks are located according to the pattern described in Figure 1, giving the lowest utilization bound among the reasonable allocation algorithms.

The reader should note that the lowest utilization bound, $L_{\mathrm{LLB}} = U_{\mathrm{wc}}^{\mathrm{LLB\text{-}WF}}$, can be applied only to steady states (a long time after the last task set change) of the system. More complex schedulability conditions are required to deal with transient states (just after the task set changes) [19].

5 Utilization bound for RAD allocation

The *Reasonable Allocation Decreasing* (RAD) algorithms are a class of reasonable allocation algorithms fulfilling the following conditions:

- Tasks are ordered by decreasing utilization factors before making the allocation, i.e, $u_1 \geq u_2 \geq \cdots \geq u_m$.
- Tasks are allocated sequentially, That is, task τ_1 is allocated first, next task τ_2, and so on until task τ_m.

The heuristics FFD and BFD, belong to this class. After ordering, the FFD heuristic allocates each task to the first processor with enough remaining capacity to hold the task. Processors are visited in the order $P_1, P_2, \ldots, P_n$. After ordering, the BFD heuristic allocates each task to the processor with the lowest remaining capacity among those with enough remaining capacity to hold the task. That is, the task is allocated to the processor in which it fits best [6].

Theorem 4 provides a lower limit on the multiprocessor utilization bound associated with the class of RAD allocation algorithms and LLB. This lower limit coincides with the upper limit on the multiprocessor utilization bound associated with any allocation algorithm and LLB. Therefore, both bounds also coincide with the utilization bound associated with any RAD allocation algorithm and LLB, as given by Corollary 2. Furthermore, RAD allocation algorithms are optimal from the point of view of the utilization bound using LLB, since there is no allocation algorithm which guarantees a higher utilization bound.

Theorem 4. *If $m > n\beta_{LLB}$, the utilization bound for RAD allocation using LLB on each processor fulfills*

$$U_{wc}^{LLB\text{-}RAD}(m, n > 1, \alpha) \geq (n\beta_{LLB} + 1)(2^{1/(\beta_{LLB}+1)} - 1)$$

Proof. Let $\{\tau_1, \ldots, \tau_m\}$ be a set of n tasks which does not fit into the multiprocessor using LLB on each processor. Let τ_k be the first task in the set which does not fit into the multiprocessor. Since RAD allocation algorithms are reasonable, from (3) we get

$$(m_j + 1)(2^{1/(m_j+1)} - 1) - U_j < u_k \qquad \text{for all } j = 1, \ldots, n \tag{14}$$

where U_j is the total utilization of the m_j tasks allocated to processor P_j, and u_k is the utilization factor of task τ_k. The total utilization of the first k tasks fulfils

$$\sum_{i=1}^{k} u_i = \sum_{j=1}^{n} U_j + u_k \tag{15}$$

From (14) and (15) we get

$$\sum_{i=1}^{k} u_i > \sum_{j=1}^{n} (m_j+1)(2^{1/(m_j+1)} - 1) - (n-1)u_k \tag{16}$$

Tasks were ordered in decreasing utilization factors before carrying out the allocation, so $u_k \leq \frac{\sum_{i=1}^{k} u_i}{k}$. Substituting this inequality into (16) and finding $\sum_{i=1}^{k} u_i$

$$\sum_{i=1}^{k} u_i > \frac{k}{k+n-1} \sum_{j=1}^{n} (m_j+1)(2^{1/(m_j+1)} - 1)$$

The total utilization of the first k tasks is less than or equal to the total utilization of the whole task set. Thus,

$$U > \frac{k}{k+n-1} \sum_{j=1}^{n} (m_j+1)(2^{1/(m_j+1)} - 1) \tag{17}$$

Let us define $I_j = (m_j + 1)$ and $g(I_1, \ldots, I_n) = \sum_{j=1}^{n} I_j (2^{1/I_j} - 1)$

$$U > \frac{k}{k+n-1} g(I_1, \ldots, I_n)$$

τ_k is the first task which does not fit into the multiprocessor. Thus, one constraint of the m_j values is that $\sum_{j=1}^{n} m_j = (k-1)$. This constraint is totally equivalent to the constraint $\sum_{j=1}^{n} I_j = (k+n-1)$. Bearing this last constraint in mind and applying Corollary 3 of Appendix with $M = (k+n-1)$

$$U > \frac{k}{k+n-1}(k+n-1)(2^{n/(k+n-1)} - 1) = k(2^{n/(k+n-1)} - 1) \tag{18}$$

The right hand term of inequality (18) increases monotonically as k increases. In addition, by definition of β_{LLB}, each processor can allocate at least β_{LLB} tasks and so $m_j \geq \beta_{\mathrm{LLB}}$. We know that $k = \sum_{j=1}^{n} m_j + 1$. Therefore, k is constrained to be $k \geq (n\beta_{\mathrm{LLB}} + 1)$. Under this constraint, the right hand term of expression (18) is minimized for $k = (n\beta_{\mathrm{LLB}} + 1)$, and so it follows that $U > (n\beta_{\mathrm{LLB}}+1)(2^{1/(\beta_{\mathrm{LLB}}+1)}-1)$. Therefore, a necessary condition to be fulfilled by the total utilization of any task set which does not fit into the n processors is $U > (n\beta_{\mathrm{LLB}}+1)(2^{1/(\beta_{\mathrm{LLB}}+1)}-1)$. In other words, any task set of total utilization less than or equal to $(n\beta_{\mathrm{LLB}}+1)(2^{1/(\beta_{\mathrm{LLB}}+1)}-1)$ fits into the n processors. Finally, we conclude $U_{\mathrm{wc}}^{\text{LLB-RAD}}(m, n > 1, \alpha) \geq (n\beta_{\mathrm{LLB}}+1)(2^{1/(\beta_{\mathrm{LLB}}+1)}-1)$ □

Corollary 2 provides the multiprocessor utilization bound for RAD allocation and LLB.

Corollary 2. *If $m > n\beta_{LLB}$*

$$U_{wc}^{\textit{LLB-RAD}}(m,n,\alpha) = \begin{cases} m(2^{1/m}-1) & \textit{if } n = 1 \\ (n\beta_{LLB}+1)(2^{1/(\beta_{LLB}+1)}-1) & \textit{if } n > 1 \end{cases}$$

Proof. The proof is direct from Theorem 2, Theorem 4, and LLB. □

6 Analysis of the theoretical results

In this section, we analyze the functions

$$L_{\mathrm{LLB}}(m,n,\alpha \leq \ln 2) = n_a U_a + n_b U_b - (n-1)\alpha$$

$$H_{\mathrm{LLB}}(m,n,\alpha) = \begin{cases} m(2^{1/m}-1) & \text{if } n = 1 \\ (n\beta_{\mathrm{LLB}}+1)(2^{1/(\beta_{\mathrm{LLB}}+1)}-1) & \text{if } n > 1 \end{cases}$$

$L_{\mathrm{LLB}}(m,n,\alpha)$ is the minimum multiprocessor utilization bound evaluated among all the reasonable allocation algorithms using LLB. This minimum coincides with the utilization bound for WF allocation. It can be seen that for the uniprocessor case $L_{\mathrm{LLB}}(m, n = 1, \alpha \leq \ln 2) = m(2^{1/m}-1)$, and so it coincides with LLB.

The expression of L_{LLB} provided in this article has one theoretical limitation: α can not be higher than $\ln 2 \approx 0.69$. However, this is not a practical limitation. For a given allocation algorithm, the utilization bound can be obtained by subtracting $\varepsilon \to 0^+$ from the minimum utilization evaluated among all the task sets which do not fit into the multiprocessor. Thus, if $\alpha > \ln 2$, all the task sets fulfilling $u_i \leq \ln 2$ also fulfill $u_i \leq \alpha$, and therefore $L_{\mathrm{LLB}}(m,n,\alpha >$

$\ln 2) \leq L_{\mathrm{LLB}}(m,n,\ln 2)$. From Figure 2, we can extrapolate the value of $L_{\mathrm{LLB}}(m,n,\ln 2)$ to be very low for $\alpha = \ln 2$, which makes the utilization bound $L_{\mathrm{LLB}}(m,n,\alpha > \ln 2)$ of little practical importance in this case.

One of the difficulties of dealing with the function $L_{\mathrm{LLB}}(m,n,\alpha \leq \ln 2)$ is its complexity. Nevertheless, if we had applied Corollary 3 of the Appendix instead of Proposition 1 in the proof of Theorem 1, we would have obtained

$$L_{\mathrm{LLB}}(m,n,\alpha) \geq (m+n-1)(2^{n/(m+n-1)}-1)-(n-1)\alpha$$

The difference $n_aU_a + n_bU_b - (m+n-1)(2^{n/(m+n-1)}-1)$ has been evaluated, giving the result

$$0 \leq \frac{n_aU_a + n_bU_b - (m+n-1)(2^{n/(m+n-1)}-1)}{n} \leq 0.0054 \quad \text{for } 1 < n < m$$

Thus, we can state $L_{\mathrm{LLB}}(m,n,\alpha \leq \ln 2) \approx (m+n-1)(2^{n/(m+n-1)}-1)-(n-1)\alpha$. In addition, it can be seen that for the uniprocessor case this expression gives LLB.

Figure 2 depicts the function $L_{\mathrm{LLB}}(m,n,\alpha \leq \ln 2)/n$ as a function of the number of processors, for different values of α. Although n is an integer, it is represented as a continuous function with the aim of improving its visualization. The representation is normalized, since L_{LLB} is divided by the number of processors, in order to show the average degree of utilization of the processors. For each value of α two different curves have been plotted. The top curve is associated with the minimum number of tasks, i.e, $m = (n\beta_{\mathrm{LLB}}+1)$. The bottom curve is associated with the maximum number of tasks, i.e, $m \to \infty$. The shaded area between the top and bottom curves corresponds to values of m in $(n\beta_{\mathrm{LLB}}+1,\infty)$.

For high values of α the utilization bound L_{LLB} is too small. However, as α nears 0, the utilization bound becomes close to $n\ln 2$. In this case, the multiprocessor behaves approximately like a uniprocessor n times faster.

$H_{\mathrm{LLB}}(m,n,\alpha)$ is the maximum of the multiprocessor utilization bounds for LLB evaluated among all the reasonable allocation algorithms. This maximum coincides with the multiprocessor utilization bound for RAD allocation. The class of RAD allocation algorithms includes allocation algorithms such as FFD and BFD. All of them have the same utilization bound, and there is no allocation algorithm which guarantees a utilization bound higher than $H_{\mathrm{LLB}}(m,n,\alpha)$

using LLB. In this respect, RAD allocation algorithms are optimal.

Figure 2 depicts the function $H_{\text{LLB}}(m,n>1,\alpha)/n$ as a function of the number of processors for different values of α. Although n is an integer, it is again represented as a continuous function to improve its visualization. This function has not been represented for $n=1$, since in this case the utilization bound coincides with the well-known utilization bound $m(2^{1/m}-1)$. For $n>1$ it does not depend on the number of tasks, m. The representation is normalized, since H_{LLB} is divided by the number of processors. Each curve in Figure 2 corresponds to a different value of β_{LLB}, and therefore to a different value of α.

For $\alpha > (2^{1/2}-1)$ we obtain $\beta_{\text{LLB}}=1$, and $H_{\text{LLB}}(m,n>1,\alpha)=(n+1)(2^{1/2}-1)$. The addition of one processor increases the value of H_{LLB} by $(2^{1/2}-1)\approx 0.414$. When $\alpha\to 0$ then $\beta_{\text{LLB}}\to\infty$ and $H(m,n>1,\alpha\to 0)=n\ln 2$. That is, the multiprocessor behaves like an ideal uniprocessor n times faster.

For example, the multiprocessor utilization bound associated with LLB and FFD allocation in a multiprocessor made up of two processors is $3(2^{1/2}-1)\approx 1.242$, about 0.62 per processor. If the tasks have utilization factors less than or equal to $(2^{1/2}-1)\approx 0.41$ then $\beta_{\text{LLB}}=2$. In this case, the utilization bound for FFD allocation takes the value $5(2^{1/3}-1)\approx 1.300$, about 0.65 per processor, close to the ideal $\ln 2\approx 0.69$.

7 Analysis of pessimism

The multiprocessor utilization bounds presented in the previous sections are useful not only to decide the schedulability of the system quickly, but also to determine the influence of key parameters on schedulability, such as the number of processors, n, the number of tasks, m, and the task sizes, α.

However, the multiprocessor utilization bounds may be too pessimistic. The pessimism of a multiprocessor utilization bound for a given allocation algorithm could be obtained by comparing this bound with an exact multiprocessor schedulability condition for the allocation algorithm. This exact multiprocessor schedulability condition consists of taking each task and trying to allocate it to any of the processors, following the allocation algorithm and using an exact uniprocessor schedulability condition on each processor to decide whether the tasks fits.

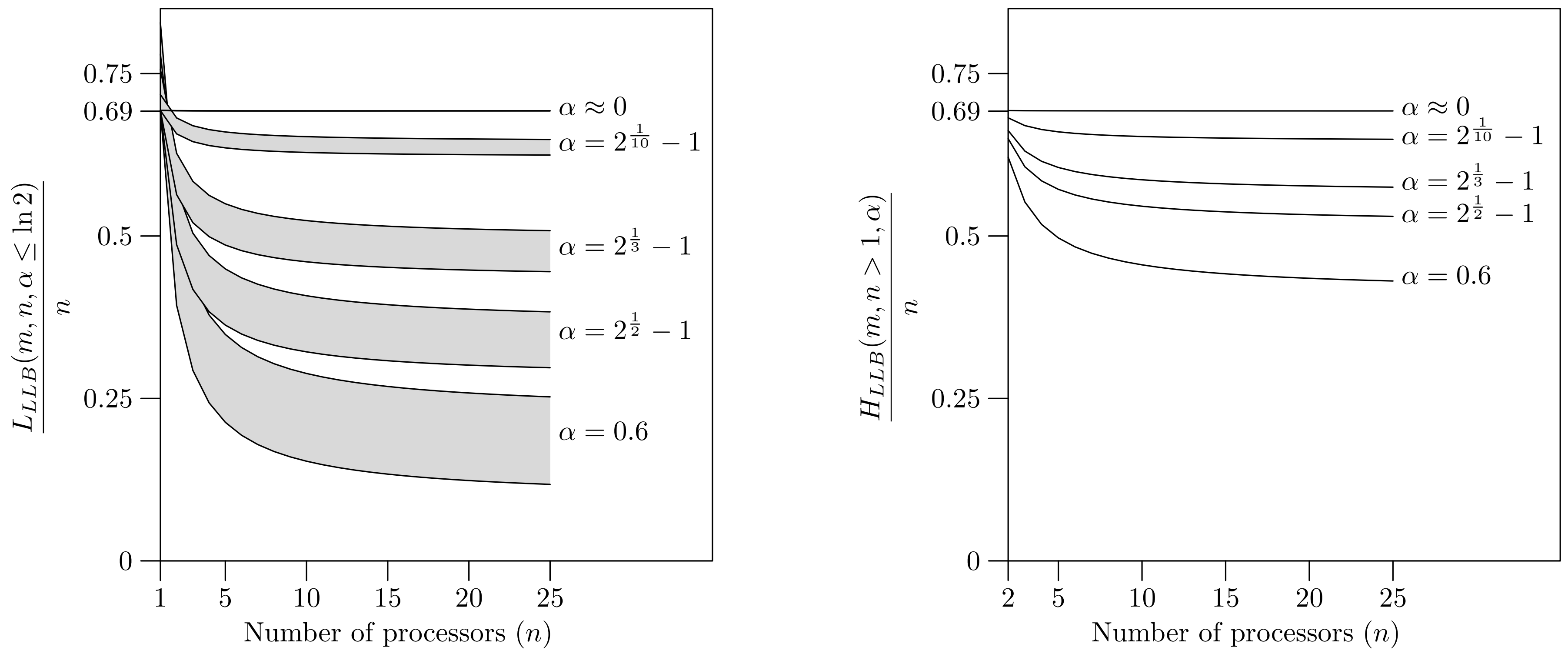

Figure 2: Plots of $L_{\text{LLB}}(m,n,\alpha \leq \ln 2)/n$ and $H_{\text{LLB}}(m,n>1,\alpha)/n$.

The task set is schedulable if all the tasks can be allocated. However, we do not follow this approach since it would require considering the task periods. Instead, we will consider a practical utilization bound of value 0.9 on each processor, as we will show later.

Extensive simulations have been carried out in order to quantify the pessimism of the multiprocessor utilization bounds. We have focused on four allocation algorithms: WF, FF, WFD and FFD. WF has been chosen because it is the worst allocation algorithm in terms of the multiprocessor utilization bound; FF because it is one of the simplest heuristics, and WFD and FFD because both provide the highest multiprocessor utilization bound and are the decreasing size counterparts of WF and FF.

Task utilization factors have been randomly generated using the Beta distribution of probability as a base. The standard deviation of the distribution, σ, is limited by a maximum value, $\sigma_{\max}$, which is a function of the expected value of the distribution. Task sets with $\alpha > \ln 2$ have been discarded, in order to meet the size limitation of the WF multiprocessor utilization bound.

Figures 3 and 4 show schedulability results for different numbers of processors and tasks, as well as for different standard deviations of the utilization factors. Each subfigure depicts six plots: three for a low value of standard deviation, $\sigma/\sigma_{\max} = 0.2$, and three for a high value of standard deviation, $\sigma/\sigma_{\max} = 0.8$. For a given standard deviation, the leftmost plot represents the percentage of schedulable task sets using the multiprocessor utilization bound derived from LLB. Note that the bound for FF allocation was presented in [15]. The central plot represents the percentage of schedulable task sets using the allocation algorithms jointly with LLB bound on each processor. That is, instead of using the multiprocessor utilization bound, an attempt is made to allocate each task of each task set following the allocation algorithm and using LLB on each processor to decide whether the task fits. The task set is schedulable if all the tasks can be allocated. This plot, when compared to the leftmost plot, allows us to estimate the pessimism of the worst-case task set used in the proof of the multiprocessor utilization bound. The rightmost plot is analogous to the central plot, but uses a uniprocessor utilization bound of value 0.9 on each processor holding two or more tasks. The value 0.9 is a practical uniprocessor utilization bound for most task sets under RM scheduling. Thus, we can approximately quantify the pessimism of using RM instead of exact uniprocessor schedulability conditions, without the need for modeling task periods, which depend largely on particular real-time systems.

We can extract several consequences from Figures 3 and 4 and additional simulations, not shown in the article:

- Schedulability depends strongly on the standard deviation of the utilization factors. If the utilization factors are very different among the tasks, schedulability becomes more difficult.
- Increasing the number of processors decreases the relative schedulability of the system. This effect was previously reported in Figure 2 for the multiprocessor utilization bounds.
- Increasing the number of tasks raises the probability of having lower values of α, improving the multiprocessor utilization bounds.
- Ordering the tasks by decreasing utilization factors makes the performance of all the allocation heuristics almost identical. WFD, FFD and BFD decreasing have almost identical behavior. The results for BFD decreasing have not been shown in the figures for the sake of brevity. This behavior coincides with that predicted from the multiprocessor utilization bounds. All the RAD allocation algorithms share a common utilization bound.

There are several factors than can contribute to the pessimism of the multiprocessor utilization bounds:

- The multiprocessor utilization bounds are obtained for worst-case task sets, which may be infrequent in practice. These worst-case task sets were defined in the proofs of Theorems 2 and 3. The pessimism coming from this source can be derived from the horizontal gap between the leftmost and central plots in the figures, for a given standard deviation. This pessimism increases with the variability of the utilization factors. Nevertheless it may be reduced by dividing the task set into several task sets, each covering a different range of utilization factors. For example, if one of the tasks has a utilization factor of 0.99 and the rest have low utilization factors, we get an improved utilization bound by adding 0.99 to the multiprocessor utilization bound for $(m-1)$ tasks and $(n-1)$ processors. A similar approach was followed in [1] to improve the relations $(N_{\mathrm{AA}}/N_{\mathrm{OPT}})$.

- Using LLB in each processor during the allocation instead of exact schedulability conditions reduces processor capacities. The capacity of one processor under RM scheduling is usually much higher than LLB [18]. Nevertheless the increment depends on task set characteristics, such as task periods and utilization factors. This pessimism for the multiprocessor schedulability is given by the horizontal gap between the central and rightmost curves, for a given standard deviation. In relative terms, this pessimism is lower for multiprocessor scheduling than for uniprocessor scheduling. The reason is that it is not always possible to take advantage of all the extra capacity.

- The multiprocessor utilization bounds derived using LLB on each processor may not be strictly tight.

8 Conclusions and future work

We have obtained the interval in which the utilization bound associated with any reasonable allocation algorithm that uses LLB is found. Since practical allocation algorithms are reasonable, the interval obtained is of wide applicability.

The WF algorithm was proved to be the worst reasonable allocation algorithm in terms of the utilization bound for multiprocessor RM scheduling using LLB. Its utilization bound is a function of the number of processors, n, the number of tasks, m, and a parameter, α, that takes the "task sizes" into account. For high values of α, the multiprocessor utilization bound is close to one for any number of processors, while for low values of α it is close to the ideal $n \ln 2$.

In addition, algorithms such as FFD and BFD were proved to be optimal in terms of the utilization bound using LLB. The utilization bound associated with these heuristics is close to the ideal $n \ln 2$ when the multiprocessor is made up of two processors, or when the utilization factors of the tasks are small. The utilization bound associated to these algorithms does not depend on the number of tasks.

Simulation results have confirmed the schedulability results derived from the theoretical multiprocessor utilization bounds. In addition, simulation results have shown that the pessimism of these bounds depends on the number of processors and the variability of the utilization factors. In general, pessimism increases when we increase these two factors.

The task set model of the article considers periodic and independent tasks. Nevertheless, it is also possible to analyze the schedulability of task sets including sporadic and aperiodic tasks, when the aperiodic tasks are served by aperiodic servers whose worst-case behavior can

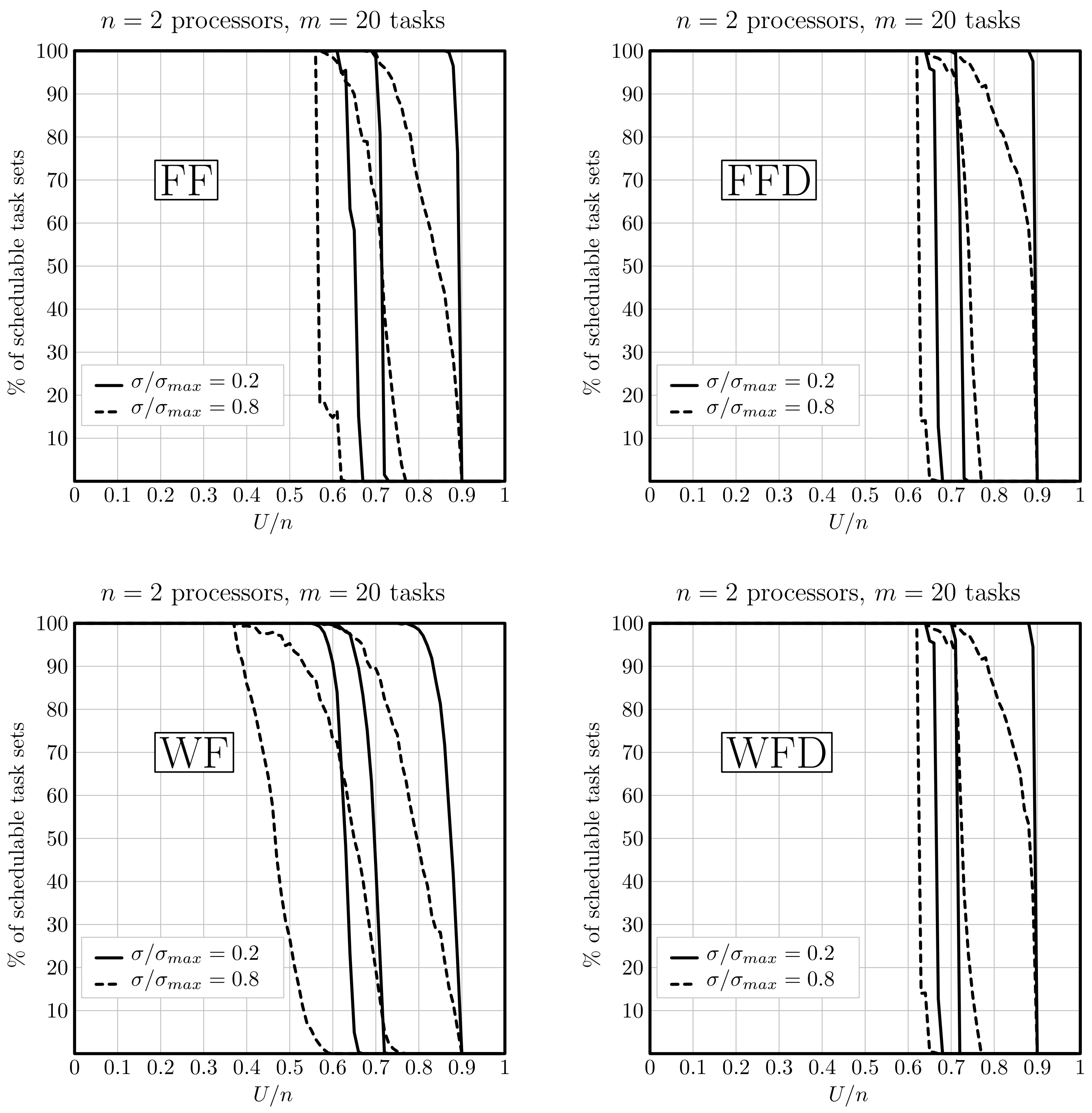

Figure 3: Simulation results for FF, FFD, WF and WFD schedulability, using $n = 2$ processors and $m = 20$ tasks.

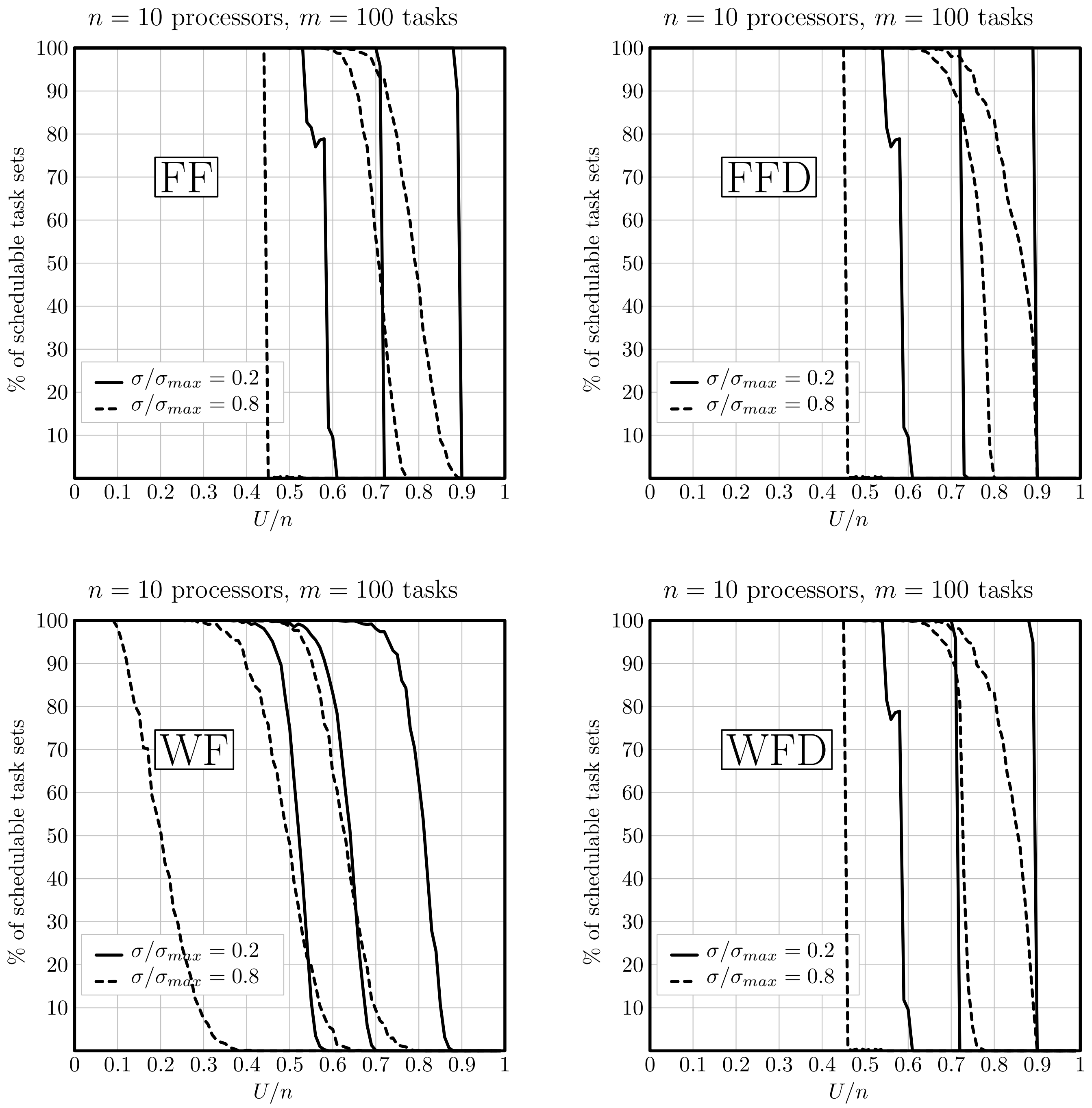

Figure 4: Simulation results for FF, FFD, WF and WFD schedulability, using $n = 10$ processors and $m = 100$ tasks.

be modeled by periodic tasks [20].

Future work will address the problem of finding the tight multiprocessor utilization bounds for multiprocessor RM scheduling. This will require the use of a necessary and sufficient schedulability condition for uniprocessor RM scheduling instead of LLB. We believe that the utilization bound provided for the class of allocation algorithms denoted by RAD is strictly tight, and that the utilization bound provided for WF allocation is almost tight. Nevertheless, this has yet to be formally proved.

Acknowledgments

Firstly, we would like to thank the reviewers. Their suggestions allow us to improve the quality of this article. Finally, we would also like to acknowledge the work of Kirstie Gough, which help us to express our ideas in English.

Appendix

Proposition 1. *Let $\{I_1,\ldots,I_n\}$ be a set of n positive natural numbers such that $\sum_{j=1}^{n} I_j = M$. Let $g(I_1,\ldots,I_n) = \sum_{j=1}^{n} I_j(2^{1/I_j}-1)$. It follows that*

$$g(I_1,\ldots,I_n) \geq (M - \lfloor M/n \rfloor n) \lceil M/n \rceil \left(2^{1/\lceil M/n \rceil}-1\right) + (n - M + \lfloor M/n \rfloor n) \lfloor M/n \rfloor \left(2^{1/\lfloor M/n \rfloor}-1\right)$$

Proof. It can be proved that the minimum of function g is obtained when M is quasi-equitably divided among $\{I_1,\ldots,I_n\}$, that is, when $|I_j - I_k| \leq 1$ for all j,k in $1,\ldots,n$. If M is a multiple of n then the quasi-equitable distribution is equitable, that is, $I_j = I_k = M/n$ for all j,k in $1,\ldots,n$. If M is not a multiple of n, the quasi-equitable distribution of M produces $(M - \lfloor M/n \rfloor n)$ terms I_j of value $\lceil M/n \rceil$, and $(n - M + \lfloor M/n \rfloor n)$ terms I_j of value $\lfloor M/n \rfloor$, i.e, one unit less. □

It is interesting to note that the right hand term of Proposition 1

$$(M - \lfloor M/n \rfloor n) \lceil M/n \rceil \left(2^{1/\lceil M/n \rceil}-1\right) + (n - M + \lfloor M/n \rfloor n) \lfloor M/n \rfloor \left(2^{1/\lfloor M/n \rfloor}-1\right)$$

decreases as M increases, since greater values of M give rise to greater elements of the vector $\{\hat{I}_1,\ldots,\hat{I}_n\}$ giving the minimum of function $\sum_{j=1}^{n} I_j(2^{1/I_j}-1)$.

Proposition 2 is analogous to Proposition 1, but instead of dividing M quasi-equitably among the natural numbers $\{I_1,\ldots,I_n\}$, it is equitably distributed among the real numbers $\{R_1,\ldots,R_n\}$.

Proposition 2. *Let $\{R_1,\ldots,R_n\}$ be a set of n positive real numbers such that $\sum_{j=1}^{n} R_j = M$. Let $g(R_1,\ldots,R_n) = \sum_{j=1}^{n} R_j(2^{1/R_j}-1)$. It follows that $g(R_1,\ldots,R_n) \geq M(2^{n/M}-1)$*

Proof. It can be proved that the minimum of function g is obtained when M is equitably divided among $\{R_1,\ldots,R_n\}$. The equitable distribution of M produces n terms R_j of value (M/n). □

Corollary 3. *Let $\{I_1,\ldots,I_n\}$ be a set of n positive natural numbers such that $\sum_{j=1}^{n} I_j = M$. Let $g(I_1,\ldots,I_n) = \sum_{j=1}^{n} I_j(2^{1/I_j}-1)$. It follows that $g(I_1,\ldots,I_n) \geq M(2^{n/M}-1)$*

Proof. The lower limit provided by Proposition 2 is applicable to the positive natural numbers since these are a subset of the positive real numbers. □

References

[1] Y. Oh and S.H. Son, “Allocating fixed-priority periodic tasks on multiprocessor systems,” *Real-Time Systems*, vol. 9, no. 3, pp. 207–239, 1995.

[2] C. L. Liu and J.W. Layland, “Scheduling algorithms for multiprogramming in a hard-real-time environment,” *Journal of the ACM*, vol. 20, no. 1, pp. 46–61, 1973.

[3] S.K. Dall and C.L. Liu, “On a real-time scheduling problem,” *Operations Research*, vol. 6, no. 1, pp. 127–140, 1978.

[4] B. Andersson, S. Baruah, and J. Jonsson, “Static-priority scheduling on multiprocessors,” in *Proceedings of the IEEE Real-Time Systems Symposium*, 2001, pp. 193–202.

[5] J. Goossens, S. Funk, and S. Baruah, “Priority-driven scheduling of periodic task systems on multiprocessors,” *Real-Time Systems*, vol. 25, no. 2-3, pp. 187–205, 2003.

[6] M.R. Garey and D.S. Johnson, *Computers and Intractability*, W.H. Freman, New York, 1979.

[7] A. Burchard, J. Liebeherr, Y. Oh, and S.H. Son, “New strategies for assigning real-time tasks to multiprocessor systems,” *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1429–1441, 1995.

[8] D. Peng, K. Shin, and T. Abdelzaher, “Assignment and scheduling communicating periodic tasks in distributed real-time systems,” *Transactions on Software Engineering*, vol. 23, no. 12, pp. 745–758, 1997.

[9] K. Tindell, A. Burns, and A. Wellings, “Allocating hard real-time tasks (an np-hard problem made easy),” *Real-Time Systems*, vol. 4, no. 2, pp. 145–165, 1992.

[10] S. Sáez, J. Vila, and A. Crespo, “Using exact feasibility tests for allocating real-time tasks in multiprocessor systems,” in *Proceedings of the 10th Euromicro Workshop on Real-Time Systems*, 1998, pp. 53–60.

[11] S. Lauzac, R. Melhem, and D. Mossé, “An efficient rms admission control and its application to multiprocessor scheduling,” in *Proceedings of the International Parallel Processing Symposium*, 1998, pp. 511–518.

[12] S. Davari and S. Dhall, “On a periodic real-time task allocation problem,” in *Annual international Conference on Systems Sciences*, 1986, pp. 133–141.

[13] S. Davari and S. Dhall, “An on line algorithm for real time tasks allocation,” in *Proceedings of the IEEE Real-Time Systems Symposium*, 1986, pp. 194–200.

[14] D. Oh and T.P. Baker, “Utilization bounds for n-processor rate monotone scheduling with static processor assignment,” *Real-Time Systems*, vol. 15, no. 2, pp. 183–193, 1998.

[15] J.M. López, M. García, J.L. Díaz, and D.F. García, “Utilization bounds for multiprocessor rate-monotonic scheduling,” *Real-Time Systems*, vol. 24, no. 1, pp. 5–28, 2003.

[16] S. K. Baruah and J. Goossens, “Rate-monotonic scheduling on uniform multiprocessors,” *IEEE Transactions on Computers*, vol. 52, no. 7, pp. 966–970, 2003.

[17] J.M. López, J.L. Díaz, M. García, and D.F. García, “Worst-case utilization bound for edf scheduling on real-time multiprocessor systems,” in *Proceedings of the Euromicro Conference on Real-Time Systems*, 2000, pp. 25–33.

[18] J. Lehoczky, L. Sha, and Y. Ding, “The rate monotonic scheduling algorithm: Exact characterization and average case behavior,” in *Proceedings of the IEEE Real-Time Systems Symposium*, 1989, pp. 166–171.

[19] P. Pedro and A. Burns, “Schedulability changes for mode changes in flexible real-time systems,” in *Proceedings of the Euromicro Workshop on Real Time Systems*, 1998, pp. 172–179.

[20] G. Bernat and A. Burns, “New results on fixed priority aperiodic servers,” in *Proceedings of the Real-Time Systems Symposium*, 1999, pp. 68–78.