# On the performance of greedy algorithms for energy minimization

Anne Benoit, Paul Renaud-Goud, Yves Robert

# On the performance of greedy algorithms for power consumption minimization

Anne Benoit, Paul Renaud-Goud, Yves Robert
*LIP, Ecole Normale Supérieure de Lyon*
*46 Allée d'Italie, 69364 Lyon Cedex 07, France*
*Email: {Anne.Benoit|Paul.Renaud-Goud|Yves.Robert}@ens-lyon.fr*

**LIP Research Report RR-LIP-2010-27, Version 2**

***Abstract*—We revisit the well-known greedy algorithm for scheduling independent jobs on parallel processors, with the objective of minimizing the power consumption. We assess the performance of the online version, as well as the performance of the offline version, which sorts the jobs by non-increasing size before execution. We derive new approximation factors, as well as examples that show that these factors cannot be improved, thereby completely characterizing the performance of the algorithms.**

***Keywords*-Greedy algorithm; Independent jobs; Parallel processors; Power consumption minimization.**

## I. Introduction

This paper aims at characterizing the performance of the well known greedy algorithm for scheduling independent jobs on parallel processors, when the objective is to minimize the power consumption instead of the execution time, or makespan.

For convenience, here is a quick background on the greedy algorithm for makespan minimization. Consider a set $\mathfrak{J}$ of $n$ independent jobs $J_1, \ldots, J_n$ to be scheduled on a set $\mathfrak{P}$ of $p$ parallel processors $\mathcal{P}_1, \ldots, \mathcal{P}_p$. Let $a_i$ be the size of job $J_i$, that is the time it requires for execution. The algorithm comes in two versions, online and offline, or without/with sorting jobs. In the online version of the problem, jobs arrive on the fly. The ONLINE-GREEDY algorithm assigns the last incoming job to the currently least loaded processor. In the offline version of the problem (see [1]), all job sizes are known in advance, and the OFFLINE-GREEDY starts by sorting the jobs (largest sizes first). Then it assigns jobs to processors exactly as in the online version. The performance of both versions is characterized by the following propositions (see Figures 1 and 2 for an illustration of the worst-case scenarios):

*Proposition 1:* For makespan minimization, ONLINE-GREEDY is a $2 - \frac{1}{p}$ approximation, and this approximation factor is met on the following instance:

- $n = p(p-1)+1$,
- $a_i = 1$ for $1 \leq i \leq n-1$,
- and $a_n = p$.

*Proposition 2:* For makespan minimization, OFFLINE-GREEDY is a $\frac{4}{3} - \frac{1}{3p}$ approximation, and this approximation factor is met on the following instance:

- $n = 2p+1$,
- $a_{2i-1} = a_{2i} = 2p-i$ for $1 \leq i \leq p$,
- and $a_n = p$.

Minimizing the total power consumed by the processors to execute the job has recently become a very important objective, both for economic and environmental reasons [2]. Assume that we can vary processor speeds, for instance through dynamic voltage scaling. In that case we can always use the smallest available speed for each processor, at the price of a dramatic decrease in performance.

The problem is in fact a bi-criteria problem: given a bound $M$ on the makespan, what is the schedule that minimizes the power consumption while enforcing the execution time bound?

For simplicity, we can assume that processors have continuous speeds (see [3], [4], [5], [6]), and scale the problem instance so that $M = 1$. This amounts to setting each processor speed equal to its workload, and to minimizing the total energy dissipated during an execution of length one time-unit. In other words, this amounts to minimizing the total dissipated power, which is proportional to the sum of the cubes of the processor speeds (a model commonly used, e.g. in [6], [7], [8], [9]).

Formally, let $alloc : \mathfrak{J} \to \mathfrak{P}$ denote the allocation function, and let $load(q) = \{i \mid alloc(J_i) = \mathcal{P}_q\}$ be the index set of jobs assigned to processor $\mathcal{P}_q$, for $1 \leq q \leq p$.

The power dissipated by $\mathcal{P}_q$ is $\left(\sum_{i \in load(q)} a_i\right)^3$, hence the objective is to minimize

$$\sum_{q=1}^{p} \left( \sum_{i \in load(q)} a_i \right)^3 . \quad (1)$$

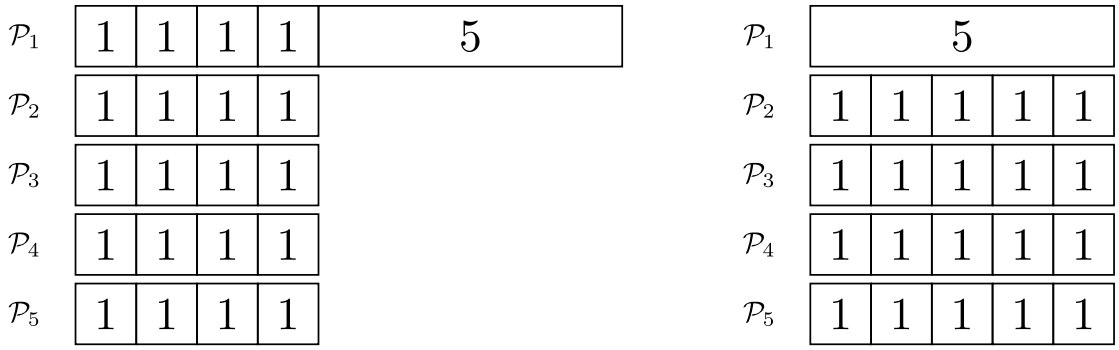


Figure 1. Tight instance for ONLINE-GREEDY (with $p = 5$).

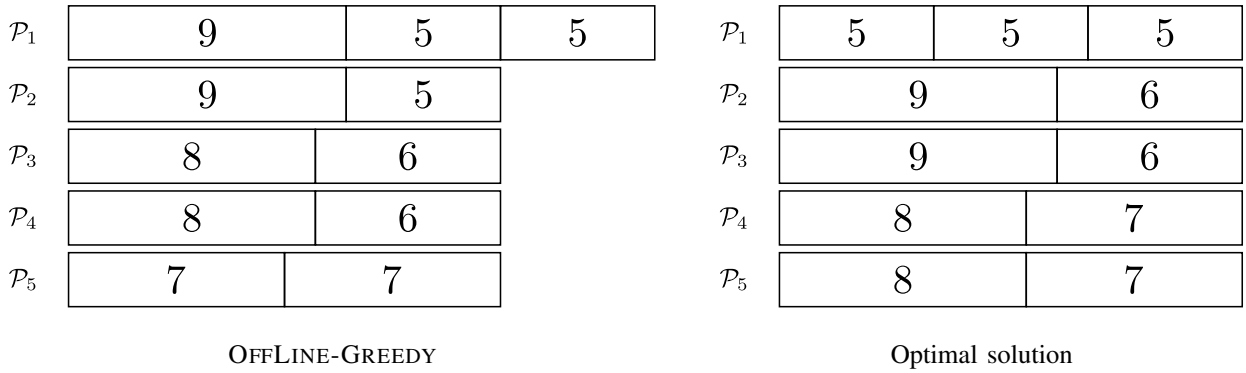


Figure 2. Tight instance for OFFLINE-GREEDY (with $p = 5$).

This is to be contrasted with the makespan minimization objective, which writes

$$\max_{1 \leq q \leq p} \sum_{i \in load(q)} a_i \ . \tag{2}$$

However, because of the convexity of the cubic power function, the "natural" greedy algorithm is the same for both objectives: assigning the next job to the currently least loaded processor minimizes, among all possible assignments for that job, both the current makespan and dissipated power. We observe that when $p = 2$, the optimal solution is the same for both objectives. However, this is not true for larger values of $p$. For example, consider the instance with $n = 6$, $p = 3$, $a_1 = 8.1$, $a_2 = a_3 = 5$, $a_4 = a_5 = 4$ and $a_6 = 2$.

- The optimal solution for the makespan is the partition $\{J_1\}, \{J_2, J_3\}, \{J_4, J_5, J_6\}$, with makespan 10 and power 2531.441.
- The optimal solution for the power is the partition $\{J_1, J_6\}, \{J_2, J_4\}, \{J_3, J_5\}$, with makespan 10.1 (hence not optimal) and power 2488.301 (the processor loads are better balanced than in the previous solution, leading to a lower power consumption).

This example is illustrated in Figure 3.

Just as the original makespan minimization problem, the (decision version of the) power minimization problem is NP-complete, and a PTAS (polynomial-time approximation scheme) can be derived. However, the greedy algorithm plays a key role in all situations where jobs arrive on the fly, or when the scheduling cost itself is critical. This was already true for the makespan problem, but may be even more important for the power problem, due to the environmental (or "green") computing perspective that applies to all application fields and computing platforms.

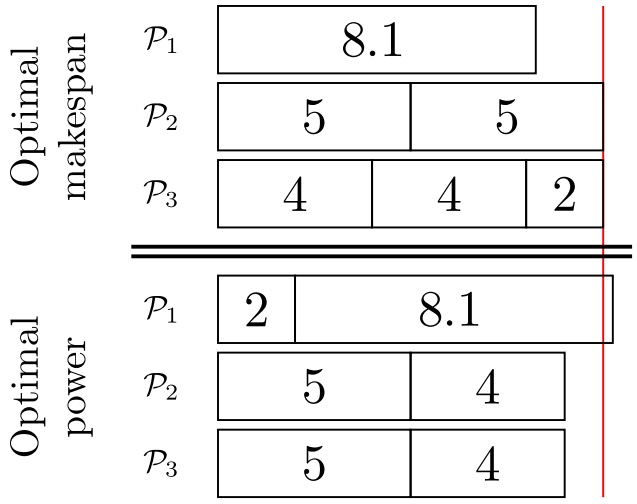


Figure 3. Different optimal solutions for makespan and power minimization.

We discuss related work in Section II. The main results of the paper are summarized in Section III, and compared to previously known results. Section IV is devoted to a detailed proof of both theorems, and also we provide in Section V numerical values of the approximation factors for small values of $p$. We give some final remarks in Section VI.

## II. Related work

The greedy algorithm has been widely studied in the literature, both in the offline and online versions. A more general problem than minimizing the sum of the cubes of the processor workloads (Equation (1)) is to minimize their $L_r$ norm, i.e., the quantity

$$N_r = \left( \sum_{q=1}^{p} \left( \sum_{i \in load(q)} a_i \right)^r \right)^{\frac{1}{r}} . \tag{3}$$

Note that

$$N_\infty = \lim_{r \to \infty} N_r = \max_{1 \le q \le p} \sum_{i \in load(q)} a_i$$

is the makespan minimization objective (Equation (2)), while $(N_3)^3$ is the power minimization objective of this paper (Equation (1)).

Chandra and Wong [10] consider the problem of minimizing $N_2$ in the offline version. They show that OffLine-Greedy is a $\frac{5}{2\sqrt{6}}$ approximation algorithm for $r = 2$, but this bound is not tight: they give lower bounds for the approximation ratio of OffLine-Greedy for the $N_2$ problem: their bound is $\frac{\sqrt{37}}{6}$ with an even number $p$ of processors, $\frac{\sqrt{83}}{9}$ with $p = 3$ processors, and $\sqrt{\frac{37}{36} - \frac{1}{36p}}$ with an odd number $p \ge 5$ of processors. The gap between these bounds has been filled by Leung and Wei [11], who provide a tight approximation factor for the performance of OffLine-Greedy for the $N_2$ problem.

Chandra and Wong [10] also provide lower and upper bounds for the approximation factor of OffLine-Greedy for the general $N_r$ problem. In particular for $r = 3$, their upper bound is $\frac{19}{45}\sqrt[3]{15} \approx 1.04$ (and their lower bound depends on the processor number $p$). Note that Theorem 2 below gives the exact approximation factor for any value of $p$, thereby closing the gap between lower and upper bounds. Finally, we point out that Chandra and Wong [10] do not deal with the online version of the problem, which Theorem 1 below completely solves.

Awerbuch et al. [12] discuss the problem of minimizing $N_r$ for general $r$ and for the online version of the problem. However, they have an additional rule: each job can be assigned only to a subset of the processors, called its *permissible* servers. They first study the problem with unit-size jobs (which is trivial without permissible servers), and they extend their analysis to the case where each job has a different execution cost on each of its admissible servers. They prove that OnLine-Greedy is a $1+\sqrt{2}$ approximation algorithm for $r = 2$, and a $\Theta(r)$ approximation algorithm in the general case.

Alon et al. [13] provide a PTAS (polynomial-time approximation scheme) to minimize $N_r$. This result is of great theoretical interest but only applies to the offline version of the problem, and is not related to the OffLine-Greedy algorithm.

Finally, Avidor et al. [14] discuss the performance of ONLINE-GREEDY when minimizing $N_r$ for general $r$. They provide an upper bound $2-\Theta(\frac{\ln r}{r})$ for the approximation factor of ONLINE-GREEDY, independently of the number of processors. This is to be contrasted with Theorem 1 which provides a tight approximation factor for any processor number in the case $r = 3$.

## III. MAIN CONTRIBUTIONS

The main results of the paper are summarized in Theorems 1 and 2 below:

*Theorem 1:* For power minimization, ONLINE-GREEDY is a $f_p^{(\mathrm{on})}(\beta_p^{(\mathrm{on})})$ approximation, where

$$f_p^{(\mathrm{on})}(\beta) = \frac{\frac{1}{p^3}\left((1+(p-1)\beta)^3 + (p-1)(1-\beta)^3\right)}{\beta^3 + \frac{(1-\beta)^3}{(p-1)^2}},$$

and where $\beta_p^{(\mathrm{on})}$ is the unique root in the interval $[\frac{1}{p}, 1]$ of the polynomial

$$\begin{aligned} g_p^{(\mathrm{on})}(\beta) = \quad & \beta^4(-p^3+4p^2-5p+2) \\ & +\beta^3(-2p^2+6p-4) \\ & +\beta^2(-4p+5) \\ & +\beta(2p-4) \\ & +1. \end{aligned}$$

This approximation factor cannot be improved.

*Theorem 2:* For power minimization, OFFLINE-GREEDY is a $f_p^{(\mathrm{off})}(\beta_p^{(\mathrm{off})})$ approximation, where

$$f_p^{(\mathrm{off})}(\beta) = \frac{\frac{1}{p^3}\left(\left(1+\frac{(p-1)\beta}{3}\right)^3 + (p-1)\left(1-\frac{\beta}{3}\right)^3\right)}{\beta^3 + \frac{(1-\beta)^3}{(p-1)^2}},$$

and where $\beta_p^{(\mathrm{off})}$ is the unique root in the interval $[\frac{1}{p}, 1]$ of the polynomial

$$\begin{aligned} g_p^{(\mathrm{off})}(\beta) = \quad & \beta^4(-9p^3+30p^2-27p+6) \\ & +\beta^3(-6p^2+18p-12) \\ & +\beta^2(-78p^2+126p+33) \\ & +\beta(18p-180) \\ & +81. \end{aligned}$$

This approximation factor cannot be improved.

We point out that this paper is the first to prove tight approximation factors for the problem of minimizing $N_3$, for any processor number $p$, both in the offline and online versions of the problem.

## IV. PROOF OF THE MAIN THEOREMS

The proof of Theorems 1 and 2 is organized as follows:

- Proposition 3 provides a technical bound that is valid for both the online and offline versions;
- This technical bound is used in Proposition 4 to show that ONLINE-GREEDY is a $f_p^{(\mathrm{on})}(\beta_p^{(\mathrm{on})})$ approximation, and in Proposition 5 to show that OFFLINE-GREEDY is a $f_p^{(\mathrm{off})}(\beta_p^{(\mathrm{off})})$ approximation;
- Finally, instances showing that the above factors are tight are given in Proposition 6 for ONLINE-GREEDY, and in Proposition 7 for OFFLINE-GREEDY.

*Proposition 3:* For any given instance, the performance ratio $\frac{P_{\mathrm{greedy}}}{P_{\mathrm{opt}}}$ of the greedy algorithm (ONLINE-GREEDY or OFFLINE-GREEDY) is such that

$$\frac{P_{\mathrm{greedy}}}{P_{\mathrm{opt}}} \leq \frac{\left(\frac{S+(p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S-a_j}{p}\right)^3}{O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3}, \tag{4}$$

where

- $P_{\mathrm{greedy}}$ is the power dissipated by the greedy algorithm;
- $P_{\mathrm{opt}}$ is the power dissipated in the optimal solution;
- $S = \sum_{i=1}^{n} a_i$;
- $O$ is the largest processor load in the optimal solution;
- $j$ is the index of the last job assigned to the processor that has the largest load in the greedy algorithm.

*Proof:* For the optimal solution, we immediately have

$$P_{\mathrm{opt}} \geq O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3.$$

This is because of the definition of $O$, and of the convexity of the power function.

There remains to show that for the greedy algorithm,

$$P_{\mathrm{greedy}} \leq \left(\frac{S+(p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S-a_j}{p}\right)^3. \tag{5}$$

Without loss of generality, let $\mathcal{P}_1$ be the maximum loaded processor in the solution returned by the greedy

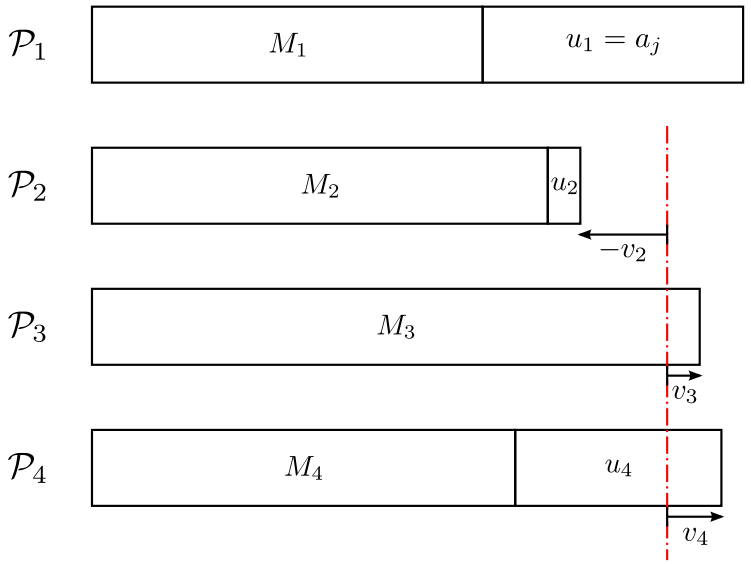


Figure 4. Notations for $p = 4$.

algorithm. For all $q \in \{1, \ldots, p\}$, let $M_q$ be the load of processor $\mathcal{P}_q$ before the assignment of the job $J_j$, and let $u_q \geq 0$ be the sum of the sizes of all jobs assigned to $\mathcal{P}_q$ after $J_{j-1}$, as illustrated in Figure 4 for $p = 4$. By definition of $j$, we have $u_1 = a_j$. In the example, $u_3 = 0$, i.e., no jobs have been assigned to $\mathcal{P}_3$ after $J_{j-1}$.

The power returned by the greedy algorithm is thus:

$$P_{\text{greedy}} = \sum_{q=1}^{p} (M_q + u_q)^3 = (M_1 + a_j)^3 + \sum_{q=2}^{p} (M_q + u_q)^3 \ .$$

For $q \in \{2, \ldots, p\}$, let $v_q$ be the variation of the load of processor $\mathcal{P}_q$ from the average load of processors other than $\mathcal{P}_1$:

$$v_q = M_q + u_q - \frac{S - M_1 - a_j}{p-1} \ ,$$

and Figure 4 illustrates this notation. Then

$$P_{\text{greedy}} = \underbrace{(M_1 + a_j)^3 + \sum_{q=2}^{p} \left( \frac{S - M_1 - a_j}{p-1} + v_q \right)^3}_{f(M_1)} .$$

Note that the $v_q$ can be either positive or negative (in the example, $v_2 \leq 0$ and $v_3 \geq 0$), and that their sum is always zero. To check this analytically, observe that $(M_1 + a_j) + \sum_{q=2}^{p} \left( \frac{S - M_1 - a_j}{p-1} + v_q \right) = S$, hence $\sum_{q=2}^{p} v_q = 0$.

Now, given the $v_q$, we have for $p \geq 2$, since $1 = \frac{p-1}{p-1}$:

$$\begin{aligned} f'(M_1) &\geq \frac{3}{p-1} \times \sum_{q=2}^{p} \left( (M_1 + a_j)^2 \right. \\ &\qquad \left. - \left( \frac{S - M_1 - a_j}{p-1} + v_q \right)^2 \right) \\ &\geq \frac{3}{p-1} \times \sum_{q=2}^{p} \left( M_1 + a_j - \frac{S - M_1 - a_j}{p-1} - v_q \right) \\ &\qquad \times \left( M_1 + a_j + \frac{S - M_1 - a_j}{p-1} + v_q \right) . \end{aligned}$$

By construction,

$$M_1 + a_j \geq \frac{S - M_1 - a_j}{p-1} + v_q ,$$

therefore $f$ is an increasing function.

Moreover, $\mathcal{P}_1$ is the least loaded processor before the assignment of $J_j$, thus a fortiori, for $q \in \{2, \ldots, p\}$,

$$M_1 \leq \frac{S - M_1 - a_j}{p-1} + v_q \ ,$$

hence

$$(p-1) M_1 \leq (S - M_1 - a_j) + \sum_{q=2}^{p} v_q = S - M_1 - a_j \ .$$

We derive that $M_1 \leq M_1^+$, where

$$M_1^+ = \frac{S - a_j}{p} \ . \tag{6}$$

Note that $M_1^+$ does not depend on the $v_q$. Since $f$ is an increasing function, we have

$$P_{\text{greedy}} = f(M_1) \leq f(M_1^+).$$

We had for $q \in \{2, \ldots, p\}$, $M_1 \leq \frac{S - M_1 - a_j}{p-1} + v_q$, hence if $M_1 = M_1^+$,

$$v_q \geq \frac{p}{p-1} \times M_1^+ - \frac{1}{p-1} \times (S - a_j) = 0.$$

We deduce that, for $M_1 = M_1^+$ and $q \in \{2, \ldots, p\}$, $v_q = 0$ (they are all nonnegative and their sum is null). Finally, we obtain

$$P_{\text{greedy}} \leq f(M_1^+) = (M_1^+ + a_j)^3 + \frac{(S - a_j - M_1^+)^3}{(p-1)^2} \ ,$$

which, given the value of $M_1^+$ from Equation (6), directly leads to Equation (5). This concludes the proof. ∎

*Proposition 4:* For power minimization, ONLINE-GREEDY is a $f_p^{(\text{on})}(\beta_p^{(\text{on})})$ approximation.

*Proof:* We use the notations of Proposition 3. We first observe that $a_i \leq O$, for all $i \in \{1, \ldots, n\}$, by definition of $O$. In particular, $a_j \leq O$.

We introduce $\beta = \frac{O}{S}$. Clearly, $\beta \in [\frac{1}{p}, 1]$, and we can rewrite Equation (4) as:

$$\frac{P_{\text{online}}}{P_{\text{opt}}} \leq \underbrace{\frac{\frac{1}{p^3}\left((1 + (p-1)\beta)^3 + (p-1)(1-\beta)^3\right)}{\beta^3 + \frac{(1-\beta)^3}{(p-1)^2}}}_{f_p^{(\text{on})}(\beta)} \ .$$

We now show that, for all $p$, $f_p^{(\text{on})}$ has a single maximum in $[\frac{1}{p}, 1]$. After differentiating with respect to $\beta$ and eliminating some positive multiplicative factor, we obtain that the sign of $\left(f_p^{(\text{on})}\right)'$ is that of $g_p^{(\text{on})}$, where:

$$g_p^{(\text{on})}(\beta) = \beta^4(-p^3 + 4p^2 - 5p + 2) + \beta^3(-2p^2 + 6p - 4) + \beta^2(-4p + 5) + \beta(2p - 4) + 1.$$

Differentiating again two times, we obtain:

$$\left(g_p^{(\text{on})}\right)'(\beta) = 4\beta^3(-p^3 + 4p^2 - 5p + 2) + 3\beta^2(-2p^2 + 6p - 4) + 2\beta(-4p + 5) + 2p - 4;$$

$$\left(g_p^{(\text{on})}\right)''(\beta) = 24\beta^2 - 24\beta + 10 - 8p + p(-12\beta p + 36\beta - 60\beta^2) + 48p^2\beta^2 - 12p^3\beta^2 \ .$$

If $p \geq 5$,

$$\begin{aligned}\left(g_p^{(\text{on})}\right)''(\beta) &\leq 34 - 40 + p(-60 + 36) \\ &\quad + p^2(-60 + 48)\beta^2 \\ &\leq 0 \ .\end{aligned}$$

We check that

$$\left(g_2^{(\text{on})}\right)''(\beta) = -6 \leq 0 \ ,$$
$$\left(g_3^{(\text{on})}\right)''(\beta) = -24\beta - 14 - 48\beta^2 \leq 0 \ \text{ and}$$
$$\left(g_4^{(\text{on})}\right)''(\beta) = -72\beta - 22 - 216\beta^2 \leq 0 \ ,$$

hence $\left(g_p^{(\text{on})}\right)'$ is a decreasing function for all $p \geq 2$ in the interval $[\frac{1}{p}, 1]$.

Next, we show that

$$\left(g_p^{(\text{on})}\right)'(1) = -4p^3 + 10p^2 - 8p + 2 \leq 0,$$

and hence either $g_p^{(\text{on})}$ is increasing and then decreasing in the interval $[\frac{1}{p}, 1]$, or $g_p^{(\text{on})}$ is decreasing in the whole interval. Indeed, for $p = 2$, $\left(g_2^{(\text{on})}\right)'(1) = -6 \leq 0$, and for $p \geq 3$, $\left(g_p^{(\text{on})}\right)'(1) \leq p^2(-12 + 10) - 24 + 2 \leq 0$.

We now check the values of $g_p^{(\text{on})}$ at the interval bounds: for $p \geq 2$, we have

$$g_p^{(\text{on})}(1) = -p + 2p^2 - p^3 \leq 0 \ , \text{ and}$$

$$g_p^{(\text{on})}\left(\tfrac{1}{p}\right) = 3 - 11/p + 15/p^2 - 9/p^3 + 2/p^4 \geq 0,$$

since $g_2^{(\text{on})}\left(\frac{1}{2}\right) = \frac{1}{4}$, $g_3^{(\text{on})}\left(\frac{1}{3}\right) = \frac{56}{81}$, and for all $p \geq 4$, $g_p^{(\text{on})}\left(\frac{1}{p}\right) \geq 3 - 11/p \geq \frac{12-11}{p} \geq 0$.

In both cases (either $g_p^{(\text{on})}$ is increasing then decreasing, or $g_p^{(\text{on})}$ is only decreasing), since $g_p^{(\text{on})}\left(\frac{1}{p}\right) \geq 0$ and $g_p^{(\text{on})}(1) \leq 0$, we conclude that $g_p^{(\text{on})}$ has a single zero $\beta_p^{(\text{on})}$ in $[\frac{1}{p}, 1]$, for which $f_p^{(\text{on})}$ attains its maximum. Finally ONLINE-GREEDY is a $f_p^{(\text{on})}(\beta_p^{(\text{on})})$ approximation. ∎

*Proposition 5:* For power minimization, OFFLINE-GREEDY is a $f_p^{(\text{off})}(\beta_p^{(\text{off})})$ approximation.

*Proof:* We follow the same line of reasoning as in Proposition 4, with $O = \beta S$, but we now further assume that $a_j \leq O/3$. Indeed, if $a_j > O/3$, there

are at most two jobs assigned to each processor in the optimal solution. But then $n \leq 2p$, and for all such instances OFFLINE-GREEDY is optimal (this is the same argument as for the makespan minimization problem, due to the convexity of the power function). With $a_j \leq O/3 = \beta S/3$, we rewrite Equation (4) as:

$$\frac{P_{\text{offline}}}{P_{\text{opt}}} \leq \underbrace{\frac{\frac{1}{p^3}\left(\left(1+\frac{(p-1)\beta}{3}\right)^3 + (p-1)\left(1-\frac{\beta}{3}\right)^3\right)}{\beta^3 + \frac{(1-\beta)^3}{(p-1)^2}}}_{f_p^{(\text{off})}(\beta)} .$$

The sign of $\left(f_p^{(\text{off})}\right)'$ is the sign of $g_p^{(\text{off})}$, where:

$$\begin{aligned} g_p^{(\text{off})}(\beta) = \quad & \beta^4(-9p^3+30p^2-27p+6) \\ & +\beta^3(-6p^2+18p-12) \\ & +\beta^2(-78p^2+126p+33) \\ & +\beta(18p-180) \\ & +81. \end{aligned}$$

Differentiating again two times, we obtain:

$$\begin{aligned} \left(g_p^{(\text{off})}\right)'(\beta) = \quad & 4\beta^3(-9p^3+30p^2-27p+6) \\ & +3\beta^2(-6p^2+18p-12) \\ & +2\beta(-78p^2+126p+33) \\ & +18p-180 ; \end{aligned}$$

$$\begin{aligned} \left(g_p^{(\text{off})}\right)''(\beta) = \quad & 12\beta^2(-9p^3+30p^2-27p+6) \\ & +6\beta(-6p^2+18p-12) \\ & -156p^2+252p+66 . \end{aligned}$$

If $p \geq 4$,

$$\begin{aligned} \left(g_p^{(\text{off})}\right)''(\beta) \leq & 12\beta^2((-36+30)p^2-108+6) \\ & +6\beta((-24+18)p-12) \\ & +(-588+252)p+(-80+66) \leq 0. \end{aligned}$$

Now $\left(g_2^{(\text{off})}\right)''(\beta) = -54$ and

$$\left(g_3^{(\text{off})}\right)''(\beta) = -576\beta^2 - 72\beta - 582 \leq 0,$$

thus for all $p > 1$ and $\frac{1}{p} \leq \beta \leq 1$, $\left(g_p^{(\text{off})}\right)''(\beta) \leq 0$. Therefore $g_p^{(\text{off})}$ is concave.

Let us now check the values of $g_p^{(\text{off})}$ at the interval bounds. We have

$$g_p^{(\text{off})}\left(\tfrac{1}{p}\right) \geq 21 - 35 + 15 \geq 0 , \text{ and}$$

$$g_p^{(\text{off})}(1) = -9p^3 - 54p^2 + 135p - 72 \leq 0 ,$$

since $g_2^{(\text{off})}(1) = -72 - 216 + 270 - 72 \leq 0$, and for $p \geq 3$, $g_p^{(\text{off})}(1) \leq p(-27 - 162 + 135) - 72 \leq 0$.

We conclude that for all $p > 1$, $f_p^{(\text{off})}$ has a single maximum in $[\frac{1}{p}, 1]$, reached for $\beta = \beta_p^{(\text{off})}$, where $g_p^{(\text{off})}(\beta_p^{(\text{off})}) = 0$. Finally OFFLINE-GREEDY is a $f_p^{(\text{off})}(\beta_p^{(\text{off})})$ approximation. ■

*Proposition 6:* The approximation factor $f_p^{(\text{on})}(\beta_p^{(\text{on})})$ for ONLINE-GREEDY cannot be improved.

*Proof:* Consider an instance with $p$ processors and $n = p(p-1)+1$ jobs, where for all $i \in \{1, \ldots, n-1\}$, $a_i = 1$, and $a_n = B = \frac{\beta_p^{(\text{on})} p(p-1)}{1-\beta_p^{(\text{on})}}$.

ONLINE-GREEDY assigns $p-1$ unit-size jobs to each processor, and then the big job is assigned to any processor, leading to a power dissipation of:

$$P_{\text{online}} = \left(\frac{S+(p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S-a_j}{p}\right)^3 ,$$

where $j = n$.

From $\beta_p^{(\text{on})} \geq \frac{1}{p}$, we deduce that $B \geq p$. Therefore the optimal solution assigns $J_n$ to the first processor, and $p$ unit-size jobs to each other processor. We have $a_j = O = B$ and for $q \in \{2, \ldots, p\}$,

$$\sum_{i \in load(q)} a_i = p = \frac{S-O}{p-1}, \text{ and hence}$$

$$P_{\text{opt}} = O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3 .$$

Moreover we have $O = \beta_p^{(\text{on})} S$:

$$\begin{aligned} O - \beta_p^{(\text{on})} S &= B - \beta_p^{(\text{on})}(B + p(p-1)) \\ &= B - p(p-1)\beta_p^{(\text{on})}\left(\frac{\beta_p^{(\text{on})}}{1-\beta_p^{(\text{on})}} + 1\right) \\ &= 0 . \end{aligned}$$

Therefore, for this instance,

$$\frac{P_{\text{online}}}{P_{\text{opt}}} = f_p^{(\text{on})}(\beta_p^{(\text{on})}) ,$$

which concludes the proof. ■

*Proposition 7:* The approximation factor $f_p^{(\text{off})}(\beta_p^{(\text{off})})$ for the ratio of the OFFLINE-GREEDY cannot be improved.

*Proof:* Consider an instance with $p$ processors and $n = 2p + 1$ jobs, where for all $1 \leq i \leq p$,

$$a_{2i-1} = a_{2i} = 2p - i + v_i,$$

and where $a_n = p + v_p$. We define

$$A = \frac{3p(1 - \beta_p^{(\text{off})} p)}{\beta_p^{(\text{off})}(p+1) - 3}, \quad \text{and}$$

$$\forall 1 \leq i \leq p, \quad v_i = \frac{i-1}{p-1} A.$$

We first show that the jobs are sorted in non-increasing order:

- For $1 \leq i \leq p$, $a_{2i-1} = a_{2i}$;
- $a_n = a_{n-1}$ $(= a_{2p})$;
- For $1 \leq i \leq p-1$, $a_{2i+1} - a_{2i} = -1 + v_{i+1} - v_i = -1 + \frac{A}{p-1}$.

Consider the function $\hat{h}_p(\beta) \mapsto \frac{3p(\beta p - 1)}{3 - \beta(p+1)}$. Its derivative is nonnegative, hence $\hat{h}_p$ is increasing.

We now prove that $\beta_p^{(\text{off})} \leq 3/(2p+1)$, which ensures that

$$A = \hat{h}_p(\beta_p^{(\text{off})}) \leq \hat{h}_p(3/(2p+1)) = p - 1,$$

and therefore $a_{2i+1} - a_{2i} = \frac{A}{p-1} - 1 \leq 0$. Recall that $\beta_p^{(\text{off})}$ is the unique root in the interval $[\frac{1}{p}, 1]$ of $g_p^{(\text{off})}$ (see Theorem 2). We already know that $g_p^{(\text{off})}(\frac{1}{p}) \geq 0$ (see proof of Proposition 5). We now prove that

$$g_p^{(\text{off})}(3/(2p+1)) \leq 0 \ .$$

Indeed, we have

$$g_p^{(\text{off})}(3/(2p+1)) = -\frac{135p(8p^3 + 3p^2 - 30p + 19)}{(2p+1)^4},$$

$$\text{and } 8p^3 + 3p^2 - 30p + 19 \geq p(32 - 30) + 19 \geq 0.$$

Therefore, $\beta_p^{(\text{off})} \leq \frac{3}{2p+1}$, which proves that $a_{2i+1} \leq a_{2i}$, and hence the jobs are sorted in non-increasing order.

Before the assignment of the last job, all processor loads are perfectly balanced. OFFLINE-GREEDY first assigns $J_1, J_2, \ldots, J_p$ to $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_p$ respectively. Then it assigns $J_{p+1}, J_{p+2}, \ldots, J_{2p}$ to $\mathcal{P}_p, \mathcal{P}_{p-1}, \ldots, \mathcal{P}_1$ respectively. After these assignments, for all $i \in \{1, \ldots, \lceil p/2 \rceil\}$, the load of processor $\mathcal{P}_{2i-1}$ is:

$$\begin{aligned} a_{2i-1} + a_{2(p-(i-1))} &= 3p + v_i + v_{p-(i-1)} \\ &= 3p + \frac{i-1}{p-1} A + \frac{p-i}{p-1} A \\ &= 3p + A. \end{aligned}$$

Moreover, for all $i \in \{1, \ldots, \lfloor p/2 \rfloor\}$, the load of processor $\mathcal{P}_{2i}$ is

$$a_{2i} + a_{2(p-(i-1))-1} = a_{2i-1} + a_{2(p-(i-1))} = 3p + A.$$

The last job $J_n$ is assigned to any processor, and the power dissipated by OFFLINE-GREEDY is:

$$P_{\text{offline}} = \left(\frac{S + (p-1)a_j}{p}\right)^3 + (p-1)\left(\frac{S - a_j}{p}\right)^3,$$

where $j = n$.

The optimal solution assigns $J_1, J_2, \ldots, J_{p-1}$ to $\mathcal{P}_2, \mathcal{P}_3, \ldots, \mathcal{P}_p$ respectively. It assigns $J_p, J_{p+1}, \ldots, J_{2p-2}$ to $\mathcal{P}_p, \mathcal{P}_{p-1}, \ldots, \mathcal{P}_2$ respectively. The last three jobs $J_{2p-1}, J_{2p}$ and $J_{2p+1}$ are assigned to $\mathcal{P}_1$, which is the most loaded processor.

The loads of processors $\mathcal{P}_2, \mathcal{P}_3, \ldots, \mathcal{P}_p$ are perfectly balanced in the optimal assignment, and their load is $3p + pA/(p-1)$:

- For all $i \in \{1, \ldots, \lfloor p/2 \rfloor\}$, the load of processor $\mathcal{P}_{2i}$ is

$$\begin{aligned} a_{2i-1} + a_{2(p-i)} &= 3p + v_i + v_{p-i} \\ &= 3p + \frac{i-1}{p-1} A + \frac{p-i+1}{p-1} A \\ &= 3p + pA/(p-1). \end{aligned}$$

- For all $i \in \{1, \ldots, \lceil p/2 \rceil - 1\}$, the load of processor $\mathcal{P}_{2i+1}$ is

$$\begin{aligned} a_{2i} + a_{2(p-i)-1} &= a_{2i-1} + a_{2(p-i)} \\ &= 3p + pA/(p-1). \end{aligned}$$

Finally, the load of processor $\mathcal{P}_1$ is

$$O = 3a_n = 3p + 3A \ ,$$

and since $3p + 3A \geq 3p + pA/(p-1)$, it is the most loaded processor.

We can then compute the corresponding power consumption. Note that the total load $S - O$ is equally divided between $p-1$ processors, and hence $3p + pA/(p-1) = \frac{S-O}{p-1}$. We obtain:

$$P_{\text{opt}} = O^3 + (p-1)\left(\frac{S-O}{p-1}\right)^3 \ .$$

| | OnLine-Greedy | | OffLine-Greedy | |
|---|---|---|---|---|
| $p$ | $\beta_p^{(\mathrm{on})}$ | $f_p^{(\mathrm{on})}(\beta_p^{(\mathrm{on})})$ | $\beta_p^{(\mathrm{off})}$ | $f_p^{(\mathrm{off})}(\beta_p^{(\mathrm{off})})$ |
| 2 | 0.577 | 1.866 | 0.513 | 1.086 |
| 3 | 0.444 | 2.008 | 0.350 | 1.081 |
| 4 | 0.372 | 2.021 | 0.267 | 1.070 |
| 5 | 0.325 | 2.001 | 0.216 | 1.061 |
| 6 | 0.292 | 1.973 | 0.181 | 1.054 |
| 7 | 0.266 | 1.943 | 0.156 | 1.048 |
| 8 | 0.246 | 1.915 | 0.137 | 1.043 |
| 64 | 0.0696 | 1.461 | 0.0177 | 1.006 |
| 512 | 0.0186 | 1.217 | 0.00223 | 1.00083 |
| 2048 | 0.00479 | 1.104 | 0.000278 | 1.00010 |
| $2^{24}$ | 0.0000192 | 1.006 | 0.0000000680 | 1.000000025 |

Table I
NUMERICAL VALUES FOR THE APPROXIMATION FACTORS OF ONLINE-GREEDY AND OFFLINE-GREEDY.

To conclude the proof, we need to prove that

$$O = \beta_p^{(\mathrm{off})} S \, .$$

Note that

$$\begin{aligned} S &= 3p^2 + v_p + 2\sum_{i=1}^{p} v_i = 3p^2 + A + \frac{2A}{p-1}\sum_{i=0}^{p-1} i \\ &= 3p^2 + (p+1)A \, , \end{aligned}$$

and therefore

$$\begin{aligned} \beta_p^{(\mathrm{off})} S - O &= 3p(\beta_p^{(\mathrm{off})} p - 1) + (\beta_p^{(\mathrm{off})}(p+1) - 3)A \\ &= 3p(\beta_p^{(\mathrm{off})} p - 1) + 3p(\beta_p^{(\mathrm{off})} p - 1) \\ &= 0, \end{aligned}$$

which leads to the desired result.

Finally, since $a_j = a_n = O/3$, we can easily verify that the ratio of this instance is

$$\frac{P_{\mathrm{offline}}}{P_{\mathrm{opt}}} = f_p^{(\mathrm{off})}(\beta_p^{(\mathrm{off})}) \, .$$

■

## V. The approximation factor as a function of $p$

We provide in this section a few observations on the values of the approximation factor of OnLine-Greedy and OffLine-Greedy for large values of $p$. Using Taylor expansions, we derive the following asymptotic values for large $p$:

- For large $p$, $\beta_p^{(\mathrm{on})} = \left(\frac{2}{p^2}\right)^{1/3} + O(1/p)$. Note that $\sqrt[3]{2} \approx 1.260$.
- For large $p$, $\beta_p^{(\mathrm{off})} = \frac{3(1+\sqrt{79})}{26p} + O(1/p^2)$. Note that $\frac{3(1+\sqrt{79})}{26} \approx 1.141$.

It is worth pointing out that both OnLine-Greedy and OffLine-Greedy are asymptotically optimal when $p$ is large, while in the case of makespan minimization, the asymptotic approximation factor of OnLine-Greedy was equal to 2 and that of OffLine-Greedy equal to $4/3$.

For $p = 2$ we have exact values: $\beta_2^{(\mathrm{on})} = \frac{\sqrt{3}}{3}$ and $f_2^{(\mathrm{on})}(\beta_2^{(\mathrm{on})}) = 1 + \frac{\sqrt{3}}{2} \approx 1.866$, while $\beta_2^{(\mathrm{off})} = \frac{\sqrt{91}-8}{3}$ and $f_2^{(\mathrm{off})}(\beta_2^{(\mathrm{off})}) = 1 + \frac{\sqrt{91}+10}{18} \approx 1.086$. We report representative numerical values in Table IV. We observe that OnLine-Greedy is at most $50\%$ more costly than the optimal for $p \geq 64$, while OffLine-Greedy always remains within $10\%$ of the optimal, and gets within $5\%$ for $p \geq 7$.

## VI. Conclusion

In this paper, we have fully characterized the performance of the greedy algorithm for the power minimization problem. We have provided tight approximation factors for any processor number $p$, both in the offline and online versions of the problem. These results extend those of a long series of papers, and completely solve the $N_3$ minimization problem.

On the practical side, further work could be devoted to conducting experiments with a more complicated power model, that would include static power in addition to dynamic power (see for example the model for the Intel Xscale [15], detailed in [16], [17], [18]). With such a model, the "natural" greedy algorithm would assign the next job to the processor that minimizes the increment in total power. There would then be two choices, either the currently least loaded processor, or a currently unused processor (at the price of more static power to be paid).

**Acknowledgments.** The authors are with Université de Lyon, France. A. Benoit and Y. Robert are with the Institut Universitaire de France. This work was supported in part by the ANR *StochaGrid* and *RESCUE* projects.

## References

[1] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal on Applied Mathematics*, vol. 17, pp. 416–429, 1969.

[2] M. P. Mills, "The internet begins with coal," *Environment and Climate News*, 1999.

[3] T. Ishihara and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," in *Proc. of International Symposium on Low Power Electronics and Design (ISLPED)*. New York, NY, USA: ACM Press, 1998, pp. 197–202.

[4] P. Langen and B. Juurlink, "Leakage-aware multiprocessor scheduling," *Journal of Signal Processing Systems*, vol. 57, no. 1, pp. 73–88, 2009.

[5] R. Mishra, N. Rastogi, D. Zhu, D. Mossé, and R. Melhem, "Energy aware scheduling for distributed real-time systems," in *Proc. of IPDPS, the Int. Parallel and Distributed Processing Symp*, 2003, pp. 21–29.

[6] K. Pruhs, R. van Stee, and P. Uthaisombut, "Speed scaling of tasks with precedence constraints," *Theory of Computing Systems*, vol. 43, pp. 67–80, 2008.

[7] A. P. Chandrakasan and A. Sinha, "JouleTrack: A Web Based Tool for Software Energy Profiling," in *Design Automation Conference*. Los Alamitos, CA, USA: IEEE Computer Society Press, 2001, pp. 220–225.

[8] H. Aydin and Q. Yang, "Energy-aware partitioning for multiprocessor real-time systems," in *Proc. of IPDPS, the Int. Parallel and Distributed Processing Symp*. IEEE CS Press, 2003, pp. 113–121.

[9] J.-J. Chen and T.-W. Kuo, "Multiprocessor energy-efficient scheduling for real-time tasks," in *Proc. of International Conference on Parallel Processing (ICPP)*. IEEE CS Press, 2005, pp. 13–20.

[10] A. K. Chandra and C. K. Wong, "Worst-case analysis of a placement algorithm related to storage allocation," *SIAM J. Computing*, vol. 4, no. 3, pp. 249–263, 1975.

[11] J. Y.-T. Leung and W.-D. Wei, "Tighter bounds on a heuristic for a partition problem," *Information Processing Letters*, vol. 56, 1995.

[12] B. Awerbuch, Y. Azar, E. Grove, M.-Y. Kao, P. Krishnan, and J. Vitter, "Load balancing in the lp norm," in *Proc. 36th Annual Symposium on Foundations of Computer Science (FOCS)*. IEEE Computer Society Press, 1995, pp. 383–391.

[13] N. Alon, Y. Azar, G. J. Woeginger, and T. Yadid, "Approximation schemes for scheduling," in *Proc. 8th ACM-SIAM Symposium on Discrete Algorithms (SODA'97)*. Society for Industrial and Applied Mathematics, 1997.

[14] A. Avidor, Y. Azar, and J. Sgall, "Ancient and new algorithms for load balancing in the lp norm," in *Proc. 9th ACM-SIAM Symposium on Discrete Algorithms (SODA'98)*. Society for Industrial and Applied Mathematics, 1998, pp. 426–435.

[15] "Intel XScale technology," http://www.intel.com/design/intelxscale.

[16] J.-J. Chen and T.-W. Kuo, "Procrastination determination for periodic real-time tasks in leakage-aware dynamic voltage scaling systems," in *Proc. of ICCAD'07, the Int. Conf. on Computer-aided design*, 2007, pp. 289–294.

[17] J.-J. Chen, "Expected energy consumption minimization in DVS systems with discrete frequencies," in *Proc. of SAC'08, Symp. on Applied Computing*, 2008, pp. 1720–1725. [Online]. Available: http://doi.acm.org/10.1145/1363686.1364095

[18] L. Niu, "Energy Efficient Scheduling for Real-Time Embedded Systems with QoS Guarantee," in *Proc. of RTCSA, the 16th Int. Conf. on Embedded and Real-Time Computing Systems and App.*, Aug. 2010, pp. 163 –172.