Resilient Scheduling of Moldable Parallel Jobs to Cope with Silent Errors
Web Supplementary Material

Anne Benoit, Valentin Le Fèvre, Lucas Perotin, Padma Raghavan, Yves Robert, Hongyang Sun

Abstract—We study the resilient scheduling of moldable parallel jobs on high-performance computing (HPC) platforms. Moldable jobs allow for choosing a processor allocation before execution, and their execution time obeys various speedup models. The objective is to minimize the overall completion time or the makespan, when jobs can fail due to silent errors and hence may need to be re-executed after each failure until successful completion. Our work generalizes the classical scheduling framework for failure-free jobs. To cope with silent errors, we introduce two resilient scheduling algorithms, LPA-LIST and BATCH-LIST, both of which use the LIST strategy to schedule the jobs. Without knowing a priori how many times each job will fail, LPA-LIST relies on a local strategy to allocate processors to the jobs, while BATCH-LIST schedules the jobs in batches and allows only a restricted number of failures per job in each batch. We prove approximation ratios for the two algorithms under several prominent speedup models (e.g., roofline, communication, Amdahl, power, monotonic, and a mix model). An extensive set of simulations is conducted to evaluate different variants of the two algorithms, and the results show that they consistently outperform some baseline heuristics. Overall, our best algorithm is within a factor of 1.6 of a lower bound on average over the entire set of experiments, and within a factor of 4.2 in the worst case.

Index Terms—Resilient scheduling, parallel jobs, moldable jobs, speedup model, failure scenario, transient errors, silent errors, list schedule, batch schedule, approximation ratios.

1 PROOF OF LEMMA 1

Lemma 1. *Given a processor allocation decision* $\mathbf{p}$ *for the jobs, the makespan of a* LIST *schedule (that determines the starting times* $\mathbf{s}$*) under any failure scenario* $\mathbf{f}$ *satisfies:*

$$T_{\text{LIST}}(\mathbf{f},\mathbf{p},\mathbf{s}) \leq \begin{cases} \frac{2A(\mathbf{f},\mathbf{p})}{P}, & \text{if } p_{\min} \geq \frac{P}{2} \\ \frac{A(\mathbf{f},\mathbf{p})}{P-p_{\min}} + \frac{(P-2p_{\min})\cdot t_{\max}(\mathbf{f},\mathbf{p})}{P-p_{\min}}, & \text{if } p_{\min} < \frac{P}{2} \end{cases}$$

where $p_{\min} \geq 1$ *denotes the minimum number of utilized processors at any time during the schedule.*

Proof. We first observe that LIST only allocates and deallocates processors upon job completions. Hence, the entire schedule can be divided into a set of consecutive and non-overlapping intervals $\mathcal{I} = \{I_1, I_2, \ldots, I_v\}$, where jobs start (or complete) at the beginning (or end) of an interval, and v denotes the total number of intervals. Let $|I_\ell|$ denote the length of interval I_ℓ. The makespan under a failure scenario $\mathbf{f}$ can then be expressed as $T_{\text{LIST}}(\mathbf{f},\mathbf{p},\mathbf{s}) = \sum_{\ell=1}^{v} |I_\ell|$.

Let $p(I_\ell)$ denote the number of utilized processors during an interval I_ℓ. Since the minimum number of utilized processors during the schedule is $p_{\min}$, we have $p(I_\ell) \geq p_{\min}$ for all $I_\ell \in \mathcal{I}$. We consider the following two cases:

Case 1: $p_{\min} \geq \frac{P}{2}$. In this case, we have $p(I_\ell) \geq p_{\min} \geq \frac{P}{2}$ for all $I_\ell \in \mathcal{I}$. Based on the definition of total cumulative area, we have $A(\mathbf{f},\mathbf{p}) = \sum_{\ell=1}^{v} |I_\ell| \cdot p(I_\ell) \geq \frac{P}{2} \cdot T_{\text{LIST}}(\mathbf{f},\mathbf{p},\mathbf{s})$. This implies that:

$$T_{\text{LIST}}(\mathbf{f},\mathbf{p},\mathbf{s}) \leq \frac{2A(\mathbf{f},\mathbf{p})}{P} .$$

Case 2: $p_{\min} < \frac{P}{2}$. Let $I_{\min}$ denote the last interval in the schedule with processor utilization $p_{\min}$, and consider a job J_j that is running during interval $I_{\min}$. Necessarily, we have $p_j \leq p_{\min}$. We now divide the set $\mathcal{I}$ of intervals into two disjoint subsets $\mathcal{I}_1$ and $\mathcal{I}_2$, where $\mathcal{I}_1$ contains the intervals in which job J_j is running (including all of its execution attempts), and $\mathcal{I}_2 = \mathcal{I} \backslash \mathcal{I}_1$. Let $T_1 = \sum_{I \in \mathcal{I}_1} |I|$ and $T_2 = \sum_{I \in \mathcal{I}_2} |I|$ denote the total lengths of all intervals in $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. Based on the definition of maximum cumulative execution time, we have $T_1 = \sum_{i=1}^{f_j+1} t_j(p_j^{(i)}) \leq t_{\max}(\mathbf{f},\mathbf{p})$.

For any interval $I \in \mathcal{I}_2$ that lies between the i-th execution attempt and the $(i+1)$-th execution attempt of J_j in the schedule, where $0 \leq i \leq f_j$, the processor utilization of I must satisfy $p(I) > P - p_{\min}$, since otherwise there are at least $p_{\min} \geq p_j$ available processors during interval I and hence the $i+1$-st execution attempt of J_j would have been scheduled at the beginning of I.

For any interval $I \in \mathcal{I}_2$ that lies after the (f_j+1)-th (last) execution attempt of J_j, there must be a job J_k running during I and that was not running during $I_{\min}$ (meaning no attempt of executing J_k was made during $I_{\min}$). This is because $p(I) > p_{\min}$, hence the job configuration must differ between I and $I_{\min}$. The processor utilization during interval I must also satisfy $p(I) > P - p_{\min}$, since otherwise the processor allocation of J_k will be $p_k \leq p(I) \leq P -$

• *A preliminary version of this work has been published in the proceedings of the IEEE Cluster'20 conference.* *Anne Benoit, Lucas Perotin and Yves Robert are with the LIP laboratory at Ecole Normale Supérieure de Lyon, France. Yves Robert is also with University of Tennessee Knoxville, USA. Valentin Le Fèvre is with Barcelona Supercomputing Center, Spain. Padma Raghavan is with Vanderbilt University, USA. Hongyang Sun is with University of Kansas, USA. Contact: anne.benoit@ens-lyon.fr*

$p_{\min}$, implying that the first execution attempt of J_k after interval $I_{\min}$ would have been scheduled at the beginning of $I_{\min}$.

Thus, for all $I \in \mathcal{I}_2$, we have $p(I) > P - p_{\min}$. Based on the definition of total cumulative area, we have $A(\mathbf{f}, \mathbf{p}) \geq (P - p_{\min}) \cdot T_2 + p_{\min} \cdot T_1$. The makespan of LIST under failure scenario $\mathbf{f}$ can then be derived as:

$$\begin{aligned} T_{\text{LIST}}(\mathbf{f}, \mathbf{p}, \mathbf{s}) &= T_1 + T_2 \\ &\leq T_1 + \frac{A(\mathbf{f}, \mathbf{p}) - p_{\min} \cdot T_1}{P - p_{\min}} \\ &= \frac{A(\mathbf{f}, \mathbf{p})}{P - p_{\min}} + \frac{(P - 2p_{\min}) \cdot T_1}{P - p_{\min}} \\ &\leq \frac{A(\mathbf{f}, \mathbf{p})}{P - p_{\min}} + \frac{(P - 2p_{\min}) \cdot t_{\max}(\mathbf{f}, \mathbf{p})}{P - p_{\min}} \ . \ \square \end{aligned}$$

2 PROOF OF LEMMA 2

Lemma 2. *Given any failure scenario* $\mathbf{f}$*, if the processor allocation decision* $\mathbf{p}$ *satisfies:*

$$\begin{aligned} A(\mathbf{f}, \mathbf{p}) &\leq \alpha \cdot A(\mathbf{f}, \mathbf{p}^*) \ , \\ t_{\max}(\mathbf{f}, \mathbf{p}) &\leq \beta \cdot t_{\max}(\mathbf{f}, \mathbf{p}^*) \ , \end{aligned}$$

where $\mathbf{p}^*$ *denotes the processor allocation of an optimal schedule, then a* LIST *schedule using processor allocation* $\mathbf{p}$ *is* $r(\alpha, \beta)$*-approximation, where*

$$r(\alpha, \beta) = \begin{cases} 2\alpha, & \text{if } \alpha \geq \beta \\ \frac{P}{P-1}\alpha + \frac{P-2}{P-1}\beta, & \text{if } \alpha < \beta \end{cases} \tag{1}$$

Proof. Based on Lemma 1, when $p_{\min} \geq \frac{P}{2}$, we have:

$$T_{\text{LIST}}(\mathbf{f}, \mathbf{p}, \mathbf{s}) \leq \frac{2A(\mathbf{f}, \mathbf{p})}{P} \leq \frac{2\alpha \cdot A(\mathbf{f}, \mathbf{p}^*)}{P} \leq 2\alpha \cdot T_{\text{OPT}}(\mathbf{f}, \mathbf{p}^*, \mathbf{s}^*).$$

The last inequality above is due to the makespan lower bound.

When $p_{\min} < \frac{P}{2}$, we can derive:

$$\begin{aligned} T_{\text{LIST}}(\mathbf{f}, \mathbf{p}, \mathbf{s}) &\leq \frac{A(\mathbf{f}, \mathbf{p})}{P - p_{\min}} + \frac{(P - 2p_{\min}) \cdot t_{\max}(\mathbf{f}, \mathbf{p})}{P - p_{\min}} \\ &\leq \frac{\alpha \cdot A(\mathbf{f}, \mathbf{p}^*)}{P - p_{\min}} + \frac{\beta(P - 2p_{\min}) \cdot t_{\max}(\mathbf{f}, \mathbf{p}^*)}{P - p_{\min}} \\ &\leq \frac{(\alpha + \beta)P - 2\beta p_{\min}}{P - p_{\min}} \cdot T_{\text{OPT}}(\mathbf{f}, \mathbf{p}^*, \mathbf{s}^*) \\ &= \Big(\alpha + \beta + (\alpha - \beta)\frac{p_{\min}}{P - p_{\min}}\Big) \cdot T_{\text{OPT}}(\mathbf{f}, \mathbf{p}^*, \mathbf{s}^*). \end{aligned}$$

We have $\frac{1}{P-1} \leq \frac{p_{\min}}{P - p_{\min}} < 1$, since $1 \leq p_{\min} < \frac{P}{2}$. Therefore, if $\alpha \geq \beta$, we get:

$$T_{\text{LIST}}(\mathbf{f}, \mathbf{p}, \mathbf{s}) \leq 2\alpha \cdot T_{\text{OPT}}(\mathbf{f}, \mathbf{p}^*, \mathbf{s}^*),$$

and if $\alpha < \beta$, we get:

$$T_{\text{LIST}}(\mathbf{f}, \mathbf{p}, \mathbf{s}) \leq \Big(\frac{P}{P-1}\alpha + \frac{P-2}{P-1}\beta\Big) \cdot T_{\text{OPT}}(\mathbf{f}, \mathbf{p}^*, \mathbf{s}^*).$$

Note that, in this case, $\frac{P}{P-1}\alpha + \frac{P-2}{P-1}\beta > 2\alpha$. $\square$

3 PROOF OF THEOREM 2

Theorem 2. LPA-LIST *is a 3-approximation for jobs with the communication model.*

Proof. In the communication model $t_j(p) = w_j/p + (p-1)c_j$, the minimum execution time $t_{\min}$ of a job J_j depends on the values of w_j and c_j. Let $p_{\min}$ denote the processor allocation that achieves $t_{\min}$, and based on the model, we get that $p_{\min} \approx \sqrt{\frac{w_j}{c_j}}$. The minimum area of the job is $a_{\min} = w_j$, achieved by allocating one processor.

We consider a processor allocation of $p_j \approx \frac{1}{2}\sqrt{\frac{w_j}{c_j}}$ for the job, and show that it achieves the bounds $\alpha = \frac{3}{2}$ and $\beta = \frac{3}{2}$, i.e., $a_j(p_j) \leq \frac{3}{2}a_{\min}$ and $t_j(p_j) \leq \frac{3}{2}t_{\min}$. Hence, based on Lemma 2, we get an approximation ratio of $2\alpha = 3$. We discuss several cases.

Case 1: If $\frac{1}{2}\sqrt{\frac{w_j}{c_j}} > P$, we set $p_j = P$.

Note that we also have $p_{\min} = P$ in this case, since $t_j(p) = w_j/p + (p-1)c_j$ is a strictly decreasing function of p in $[1, P]$. Thus, we have $t_j(p_j) = \frac{w_j}{P} + (P-1)c_j = t_{\min}$. The area of the job satisfies $a_j(p_j) = w_j + P(P-1)c_j \leq w_j + P^2 c_j < w_j + \frac{1}{4}w_j = \frac{5}{4}a_{\min}$.

Case 2: If $\frac{1}{2}\sqrt{\frac{w_j}{c_j}} < \frac{3}{2}$, then the minimum of $t_j(p)$ is achieved at $p^* = \sqrt{\frac{w_j}{c_j}} \in (0, 3)$. We consider three subcases depending on the values of p^* and/or $p_{\min}$.

Case 2.1: If $p_{\min} = 1$, we set $p_j = 1$.

In this case, we must have $p^* \in (0, 2]$. Therefore, $t_j(p_j) = t_{\min}$ and $a_j(p_j) = a_{\min}$.

Case 2.2: If $p^* \in (1, 2]$ and $p_{\min} = 2$, we set $p_j = 1$.

In this case, since $p^* = \sqrt{\frac{w_j}{c_j}} \leq 2$, we have $c_j \geq \frac{1}{4}w_j$. The area of the job using $p_j = 1$ is $a_j(p_j) = a_{\min}$. The minimum execution time of the job is $t_{\min} = \frac{w_j}{2} + c_j \geq \frac{3}{4}w_j$, and the execution time using $p_j = 1$ satisfies $t_j(p_j) = w_j \leq \frac{4}{3}t_{\min}$.

Case 2.3: If $p^* \in (2, 3)$, we set $p_j = 2$.

In this case, since $p^* = \sqrt{\frac{w_j}{c_j}} > 2$, we have $c_j < \frac{1}{4}w_j$. Also, we must have $p_{\min} = 2$ or 3. The area of the job using $p_j = 2$ is $a_j(p_j) = w_j + p_j(p_j - 1)c_j = w_j + 2c_j < \frac{3}{2}w_j = \frac{3}{2}a_{\min}$. The minimum execution time of the job satisfies $t_{\min} = \frac{w_j}{p_{\min}} + (p_{\min} - 1)c_j \geq \frac{w_j}{3} + c_j$, and the execution time using $p_j = 2$ satisfies $t_j(p_j) = \frac{w_j}{2} + c_j \leq \frac{3}{2}(\frac{w_j}{3} + c_j) \leq \frac{3}{2}t_{\min}$.

Case 3: If $\frac{3}{2} \leq \frac{1}{2}\sqrt{\frac{w_j}{c_j}} \leq P$, then we have $c_j \leq \frac{1}{9}w_j$. The minimum of $t_j(p)$ is achieved at $p^* = \sqrt{\frac{w_j}{c_j}}$. Thus, the minimum execution time of the job satisfies $t_{\min} \geq t_j(p^*) = 2\sqrt{w_j c_j} - c_j$.

Let $\frac{1}{2}\sqrt{\frac{w_j}{c_j}} = q + r$, where q denotes the largest integer such that $q \leq \frac{1}{2}\sqrt{\frac{w_j}{c_j}}$ and $r = \frac{1}{2}\sqrt{\frac{w_j}{c_j}} - q$ denotes the remaining fraction. We set p_j by rounding $\frac{1}{2}\sqrt{\frac{w_j}{c_j}}$ as follows:

$$p_j = \begin{cases} q, & \text{if } r \leq 0.2 \\ q + 1, & \text{if } r > 0.2 \end{cases} \tag{2}$$

and we consider two subcases depending on the value of p_j.

Case 3.1: If $p_j = q$, we have $\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 0.2 \leq p_j \leq \frac{1}{2}\sqrt{\frac{w_j}{c_j}}$. The area of the job using p_j is $a_j(p_j) = w_j + p_j(p_j - 1)c_j \leq$

$w_j + p_j^2 c_j \le \frac{5}{4} w_j = \frac{5}{4} a_{\min}$. The execution time of the job using p_j satisfies:

$$\begin{aligned} t_j(p_j) &= \frac{w_j}{p_j} + (p_j - 1)c_j \\ &\le \frac{w_j}{\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 0.2} + \Big(\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 1\Big)c_j \\ &\le \frac{3}{2}\Big(2\sqrt{w_j c_j} - c_j\Big) \le \frac{3}{2} t_{\min} \,. \end{aligned}$$

The second last inequality above is shown below:

$$\begin{aligned} & \frac{w_j}{\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 0.2} + \Big(\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 1\Big)c_j \le \frac{3}{2}\Big(2\sqrt{w_j c_j} - c_j\Big) \\ \Leftarrow \quad & \frac{w_j}{\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 0.2} \le \frac{5}{2}\sqrt{w_j c_j} - \frac{1}{2} c_j \\ \Leftarrow \quad & w_j \le \Big(\frac{5}{2}\sqrt{w_j c_j} - \frac{1}{2} c_j\Big)\Big(\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 0.2\Big) \\ \Leftarrow \quad & \frac{3}{4}\sqrt{w_j c_j} \le \frac{1}{4} w_j + \frac{1}{10} c_j \\ \Leftarrow \quad & \frac{3}{4}\sqrt{w_j c_j} \le \frac{1}{4} w_j \\ \Leftarrow \quad & c_j \le \frac{1}{9} w_j \end{aligned}$$

Case 3.2: If $p_j = q + 1$, we have $\frac{1}{2}\sqrt{\frac{w_j}{c_j}} \le p_j \le \frac{1}{2}\sqrt{\frac{w_j}{c_j}} + 0.8$. The area of the job using p_j satisfies:

$$\begin{aligned} a_j(p_j) &= w_j + p_j(p_j - 1)c_j \\ &\le w_j + \Big(\frac{1}{2}\sqrt{\frac{w_j}{c_j}} + 0.8\Big)\Big(\frac{1}{2}\sqrt{\frac{w_j}{c_j}} - 0.2\Big)c_j \\ &= w_j + \Big(\frac{w_j}{4c_j} + \frac{3}{10}\sqrt{\frac{w_j}{c_j}} - 0.16\Big)c_j \\ &\le \frac{5}{4} w_j + \frac{3}{10}\sqrt{w_j c_j} \\ &\le \frac{5}{4} w_j + \frac{1}{10} w_j < \frac{3}{2} a_{\min} \,. \end{aligned}$$

The second last inequality above is because of $c_j \le \frac{1}{9} w_j$.

The execution time of the job using p_j satisfies:

$$\begin{aligned} t_j(p_j) &= \frac{w_j}{p_j} + (p_j - 1)c_j \\ &\le \frac{w_j}{\frac{1}{2}\sqrt{\frac{w_j}{c_j}}} + \Big(\frac{1}{2}\sqrt{\frac{w_j}{c_j}} + 0.8 - 1\Big)c_j \\ &\le \frac{5}{2}\sqrt{w_j c_j} - 0.2 c_j \\ &\le \frac{3}{2}\Big(2\sqrt{w_j c_j} - c_j\Big) \le \frac{3}{2} t_{\min} \,. \end{aligned}$$

The second last inequality above is shown below:

$$\begin{aligned} & \frac{5}{2}\sqrt{w_j c_j} - 0.2 c_j \le \frac{3}{2}\Big(2\sqrt{w_j c_j} - c_j\Big) \\ \Leftarrow \quad & \frac{13}{10} c_j \le \frac{1}{2}\sqrt{w_j c_j} \\ \Leftarrow \quad & c_j \le \Big(\frac{5}{13}\Big)^2 w_j \\ \Leftarrow \quad & c_j \le \frac{1}{9} w_j \end{aligned}$$

Thus, in all cases, we have shown $a_j(p_j) \le \frac{3}{2} a_{\min}$ and $t_j(p_j) \le \frac{3}{2} t_{\min}$. This completes the proof of the theorem. □

4 Proof of Theorem 3

Theorem 3. *The approximation ratio of* LPA-LIST *is at least 2.5 for jobs with the communication model.*

Proof. We consider $n = 2K$ identical jobs to be processed on $P = 2K$ processors for some $K > 2$. Suppose $w = (2K+2)^2$ and $c = 1$ for all jobs and that the jobs do not fail. We will show the following under these settings:

1) If $p \le K + 1$, $\alpha(p) \le \beta(p)$;
2) If $p \ge K + 2$, $\alpha(p) \ge \beta(p)$;
3) When K is large enough, $r(\alpha(p), \beta(p))$ is decreasing with p in $[1, K+1]$ and increasing with p in $[K+2, 2K]$;
4) When K is large enough, LPA-LIST will allocate $p^* \in \{K+1, K+2\}$ processors to each job;
5) When K approaches infinity, the approximation ratio approaches 2.5.

First, $t(p) = \frac{w}{p} + p - 1$ decreases for $p \in [1, \sqrt{w}]$. As $\sqrt{w} > P$, we have $t_{\min} = t(P) = \frac{(2K+2)^2}{2K} + 2K - 1 = 4K + 3 + \frac{2}{K}$. This also shows that $t(p)$ is (strictly) decreasing with p, and so is $\beta(p)$, while $a(p) = w + p(p-1)$ is (strictly) increasing with p, and so is $\alpha(p)$. This means there exists at most one value $\bar{p}$ that satisfies $\alpha(\bar{p}) = \beta(\bar{p})$, and if it exists, then $p \le \bar{p} \Leftrightarrow \alpha(p) \le \beta(p)$. We will show that such $\bar{p}$ exists and is in $[K+1, K+2]$, which proves the first two points.

$$\begin{aligned} \alpha(\bar{p}) = \beta(\bar{p}) &\Leftrightarrow \frac{a(\bar{p})}{a_{\min}} = \frac{t(\bar{p})}{t_{\min}} = \frac{\bar{p} t(\bar{p})}{\bar{p} t_{\min}} = \frac{a(\bar{p})}{\bar{p} t_{\min}} \\ &\Leftrightarrow \bar{p} = \frac{a_{\min}}{t_{\min}} \\ &\Leftrightarrow \bar{p} = \frac{K(2K+2)^2}{4K^2 + 3K + 2} \\ &\Leftrightarrow \bar{p} = K + 1 + \frac{K^2 + K - 2}{4K^2 + 3K + 2} \end{aligned}$$

The above shows $\bar{p} \in [K+1, K+2]$. Therefore, when $p \ge K + 2$, we have $\alpha(p) \ge \beta(p)$, thus based on Lemma 2, $r(\alpha(p), \beta(p)) = 2\alpha(p)$, which is clearly increasing with p in $[K+2, 2K+1]$. On the other hand, when $p \le K + 1$, we have $\alpha(p) \le \beta(p)$, so $r(\alpha(p), \beta(p)) = \frac{P}{P-1}\alpha(p) + \frac{P-2}{P-1}\beta(p) \triangleq f(p)$. In the following, we will show that, when K is large enough, $f(p)$ is decreasing with p in $[1, K+1]$, which proves the third point.

$$\begin{aligned} f(p) &= \frac{P}{P-1}\frac{t(p)p}{a_{\min}} + \frac{P-2}{P-1}\frac{t(p)}{t_{\min}} \\ &= \frac{2K}{2K-1}\frac{(2K+2)^2 + p(p-1)}{(2K+2)^2} \\ &\quad + \frac{2K-2}{(2K-1)t_{\min}}\left(\frac{(2K+2)^2}{p} + p - 1\right) \,, \\ f'(p) &= \frac{2K(2p-1)}{(2K-1)(2K+2)^2} + \frac{2K-2}{(2K-1)t_{\min}}\left(1 - \frac{(2K+2)^2}{p^2}\right) \,. \end{aligned}$$

We can see that $f'(p)$ is clearly increasing with p, so $f(p)$ will be decreasing with p in $[1, K+1]$ if $f'(K+1) < 0$, which is true when K is large enough as shown below.

$$\begin{aligned} f'(K+1) < 0 &\Leftrightarrow (2K-1)f'(K+1) < 0 \\ &\Leftrightarrow \frac{2K(2K+1)}{(2K+2)^2} + \frac{2K-2}{4K+3+\frac{2}{K}} \\ &\quad < \frac{(2K-2)(2K+2)^2}{(4K+3+\frac{2}{K})(K+1)^2} \end{aligned}$$

In the last inequality above, the left-hand side is equivalent to $\frac{4K^2}{4K^2}+\frac{2K}{4K}$ when K approaches infinity thus converges to 1.5, whereas the right-hand side is equivalent to $\frac{8K^3}{4K^3}$ thus converges to 2. This means that, when K is large enough, $f'(K+1) < 0$ and thus $r(\alpha(p), \beta(p))$ is decreasing with p in $[1, K+1]$. All together, we can conclude that the smallest ratio $r(\alpha(p), \beta(p))$ is achieved with $p^* \in \{K+1, K+2\}$, which proves the fourth point.

Given the results above, we can now estimate the makespan of LPA-LIST. Since at least $K+1$ processors will be allocated to each job, no job may be processed in parallel, and the execution time of each job will be at least $\min(t(K+2), t(K+1)) = t(K+2) = \frac{(2K+2)^2}{K+2} + K + 1$. Thus, the makespan of LPA-LIST satisfies:

$$T \geq 2K\left(\frac{(2K+2)^2}{K+2} + K + 1\right) = \frac{10K^3 + 22K^2 + 12K}{K+2} ,$$

which is equivalent to $10K^2$ when K approaches infinity. The optimal algorithm, on the other hand, would schedule each job on a single processor, resulting in a makespan of $T_{\text{OPT}} = (2K+2)^2$, which is equivalent to $4K^2$ when K approaches infinity. This shows that, for any $\epsilon > 0$, $\frac{T}{T_{\text{OPT}}}$ can be larger than $2.5 - \epsilon$, and thus LPA-LIST is at least a 2.5-approximation for jobs with the communication model. □

5 PROOF OF THEOREM 5

Theorem 5. *The approximation ratio of* LPA-LIST *is at least* 3 *for jobs with the Amdahl's model.*

Proof. We consider $n = 2K+1$ identical jobs to be processed on $P = 2K+1$ processors for some $K > 2$. Suppose $w = 2K+1$ and $d = 1$ for all jobs and that the jobs do not fail. Under these settings, we have $t_{\min} = 2$ and $a_{\min} = 2K+2$, thus $\alpha(p) = \frac{w+dp}{a_{\min}} = \frac{2K+1+p}{2K+2}$ and $\beta(p) = \frac{\frac{w}{p}+d}{t_{\min}} = \frac{2K+1+p}{2p}$. Clearly, $\alpha(p)$ strictly increases with p while $\beta(p)$ strictly decreases with p. Further, there exists a unique p^* that satisfies $\alpha(p^*) = \beta(p^*)$, as shown below:

$$\begin{aligned}\alpha(p^*) = \beta(p^*) &\Leftrightarrow 2p^*(2K+1+p^*) = (2K+2)(2K+1+p^*)\\ &\Leftrightarrow p^* = K+1\end{aligned}$$

We will show that $p^* = K+1$ minimizes $r(\alpha(p), \beta(p))$. To that end, we first notice that, when $p \geq p^*$, we have $\alpha(p) \geq \beta(p)$, so based on Lemma 2, $r(\alpha(p), \beta(p)) = 2\alpha(p)$, which is clearly increasing with p in $[p^*, P]$. Otherwise, when $p \leq p^*$, we have $\alpha(p) \leq \beta(p)$, so $r(\alpha(p), \beta(p)) = \frac{P}{P-1}\alpha(p) + \frac{P-2}{P-1}\beta(p) \triangleq f(p)$. We will show that, for $K > 2$, $f(p)$ is decreasing with p in $[1, p^*]$. Altogether, these results will show that p^* minimizes $r(\alpha(p), \beta(p))$, hence will be chosen by the LPA-LIST algorithm.

Now, to show that $f(p)$ is decreasing with p in $[1, p^*]$, we plug in $P = 2K+1$ and obtain:

$$f(p) = \frac{2K+1}{2K} \cdot \frac{2K+1+p}{2K+2} + \frac{2K-1}{2K}\left(\frac{1}{2} + \frac{2K+1}{2p}\right) ,$$

$$f'(p) = \frac{2K+1}{4K(K+1)} - \frac{(2K-1)(2K+1)}{4Kp^2} .$$

We can see that $f'(p)$ is increasing with p, so $f(p)$ will be decreasing with p in $[1, p^*]$ if $f'(p^*) < 0$, which is true when $K > 2$ as shown below.

$$\begin{aligned}f'(p^*) &= \frac{2K+1}{4K(K+1)} - \frac{(2K-1)(2K+1)}{4K(K+1)^2}\\ &= \frac{(2K+1)(2-K)}{4K(K+1)^2} < 0 .\end{aligned}$$

Thus, LPA-LIST will allocate $p^* = K+1$ processors to each job. Since there are $P = 2K+1$ processors in total, all jobs will be executed sequentially and the makespan will be $T = (2K+1)\left(\frac{2K+1}{K+1} + 1\right) = \frac{(2K+1)(3K+2)}{K+1}$.

However, the optimal algorithm would allocate one processor to each job, resulting in a makespan of $T_{\text{OPT}} = 2K+2$. Thus, the approximation ratio is:

$$\begin{aligned}\frac{T}{T_{\text{OPT}}} &= \frac{(2K+1)(3K+2)}{(K+1)(2K+2)}\\ &= \frac{3K+2}{K+1} - \frac{3K+2}{(K+1)(2K+2)}\\ &= 3 - \frac{1}{K+1} - \frac{3K+2}{(K+1)(2K+2)} .\end{aligned}$$

When K is large enough, the above ratio can be larger than $3-\epsilon$ for any $\epsilon > 0$. This shows that LPA-LIST is at least a 3-approximation for jobs with the Amdahl's model. □

6 PROOF OF THEOREM 6

Theorem 6. LPA-LIST *is a 6-approximation for jobs with the mix model.*

Proof. In the mix model with $t_j(p) = c_j\left(\frac{w'_j}{p} + d'_j + (p-1)\right)$, we provide a processor allocation p_j for a job J_j and show that it achieves the bounds $\alpha = \beta = 3$, i.e., $a_j(p_j) \leq 3a_{\min}$ and $t_j(p_j) \leq 3t_{\min}$. Hence, based on Lemma 2, we get an approximation ratio of $2\alpha = 6$. We discuss two cases.

Case 1: $w'_j \leq 1$. In this case, both execution time and area are increasing with the processor allocation, so allocating $p_j = 1$ processor gives the optimal time and area.

Case 2: $w'_j > 1$. In this case, the execution time of a job J_j is minimized at $p = \sqrt{w'_j}$, which gives $t_{\min} \geq c_j(2\sqrt{w_j} + d'_j - 1) > c_j(\sqrt{w'_j} + d'_j)$. The minimum area of the job is $a_{\min} = c_j(w'_j + d'_j)$, achieved by allocating one processor. We consider two sub-cases depending on the value of $\bar{p}_j$.

Case 2.1: $\bar{p}_j \geq \left\lceil \frac{w'_j + d'_j}{\sqrt{w'_j} + d'_j} \right\rceil$. In this case, we consider a processor allocation $p'_j = \left\lceil \frac{w'_j + d'_j}{\sqrt{w'_j} + d'_j} \right\rceil$. With this choice, we have $p'_j t_{\min} \geq a_{\min}$, which implies $\frac{t_j(p'_j)}{t_{\min}} \leq \frac{a_j(p'_j)}{a_{\min}}$. We will now upper bound the latter:

$$\frac{a_j(p'_j)}{a_{\min}} = \frac{w'_j + p'_j(d'_j + p'_j - 1)}{w'_j + d'_j}$$
$$< 3 \cdot \frac{w'_j + \left(\frac{w'_j + d'_j}{\sqrt{w'_j + d'_j}} + 1\right)\left(d'_j + \frac{w'_j + d'_j}{\sqrt{w'_j + d'_j}}\right)}{w'_j + 2(w'_j + d'_j)}$$
$$< 3 \cdot \frac{w'_j \left(\sqrt{w'_j} + d'_j\right)^2}{w'_j \left(\sqrt{w'_j} + d'_j\right)^2 + 2(w'_j + d'_j)\left(\sqrt{w'_j} + d'_j\right)^2}$$
$$+ 3 \cdot \frac{\left(w'_j + d'_j + \sqrt{w'_j} + d'_j\right)\left(d'_j\left(\sqrt{w'_j} + d'_j\right) + w'_j + d'_j\right)}{w'_j \left(\sqrt{w'_j} + d'_j\right)^2 + 2(w'_j + d'_j)\left(\sqrt{w'_j} + d'_j\right)^2}$$
$$< 3 \cdot \frac{w'_j \left(\sqrt{w'_j} + d'_j\right)^2 + 2(w'_j + d'_j)\left(w'_j + 2d'_j\sqrt{w'_j} + {d'_j}^2\right)}{w'_j \left(\sqrt{w'_j} + d'_j\right)^2 + 2\left(w'_j + d'_j\right)\left(w'_j + 2d'_j\sqrt{w'_j} + {d'_j}^2\right)}$$
$$= 3 \, .$$

The last inequality is obtained by applying $\sqrt{w'_j} < w'_j$ and $1 < \sqrt{w'_j}$ on top. Thus, we get $\frac{t_j(p'_j)}{t_{\min}} \leq \frac{a_j(p'_j)}{a_{\min}} < 3$.

Case 2.2: $\bar{p}_j < \left\lceil \frac{w'_j + d'_j}{\sqrt{w'_j + d'_j}} \right\rceil$. In this case, we consider a processor allocation p''_j that minimizes $t_j(p)$, i.e., $t_j(p''_j) = t_{\min}$. Since $p''_j \leq \bar{p}_j < p'_j$ and $a_j(p)$ is increasing with p, we have $a_j(p''_j) < a_j(p'_j) < 3a_{\min}$, as shown in Case 2.1. □

7 Proof of Theorem 13

Theorem 13. *The expected approximation ratio of* BATCH-LIST *is $\omega(1)$, if all jobs have constant failure probabilities.*

Before proving the theorem, we first compute the probability that BATCH-LIST produces exactly b batches.

Lemma 3. *The probability that there are exactly b batches in a* BATCH-LIST *schedule, where $b \geq 1$, is given by:*

$$Q_b = \prod_{j=1}^{n}(1 - q_j^{2^b - 1}) - \prod_{j=1}^{n}(1 - q_j^{2^{b-1} - 1}) \, .$$

Proof. For any $b \geq 0$, let R_b denote the probability that there are at most b batches in the schedule. According to the BATCH-LIST algorithm, this happens when the number of failures f_j of any job J_j satisfies $f_j \leq 2^b - 2$, for all $1 \leq j \leq n$. Thus, we can compute R_b as follows:

$$\begin{aligned} R_b &= \prod_{j=1}^{n} \mathbb{P}(f_j \leq 2^b - 2) \\ &= \prod_{j=1}^{n} \sum_{k=0}^{2^b - 2} \mathbb{P}(f_j = k) \\ &= \prod_{j=1}^{n} \sum_{k=0}^{2^b - 2} (1 - q_j) q_j^k \\ &= \prod_{j=1}^{n} (1 - q_j^{2^b - 1}) \, . \end{aligned}$$

The probability that there are exactly b batches is therefore given by $Q_b = R_b - R_{b-1}$, for any $b \geq 1$. □

(Proof of Theorem 13). To prove the claim, we show that, for any given constant $C > 0$, there exists an instance such that the expected approximation ratio of the BATCH-LIST algorithm is strictly larger than C.

We construct the instance similarly to the one in the proof of Theorem 9. Specifically, we consider a set $\mathcal{J} = \{J_1, J_2, \ldots, J_K\}$ of K sequential jobs and at least as many processors, so that each job can be executed on a dedicated processor. For each job J_j, where $1 \leq j \leq K$, its (sequential) execution time is given by $t_j = \frac{1}{2^j}$ and its failure probability q_j is defined arbitrarily but upper-bounded by a constant $\rho < 1$.

Consider a failure scenario $\mathbf{f}$, in which each job J_j fails until batch B_{K+j} where it finally completes successfully. Hence, the total number of execution attempts of job J_j is at most 2^{K+j}, and the time to complete the job is at most $2^{K+j} \cdot \frac{1}{2^j} = 2^K$. The optimal makespan for this failure scenario therefore satisfies $T_{\text{OPT}}(\mathcal{J}, \mathbf{f}) \leq 2^K$.

Consider the BATCH-LIST algorithm under the same failure scenario $\mathbf{f}$. In each batch B_{K+j}, where $1 \leq j \leq K-1$, job J_{j+1} does not complete successfully and is thus executed 2^{K+j-1} times. The execution time of this batch is therefore at least $2^{K+j-1} \cdot \frac{1}{2^{(j+1)}} = 2^{K-2}$. The total time to complete batches B_{K+1} to B_{2K-1}, and hence the makespan of BATCH-LIST, is at least $T_{\text{BATCH-LIST}}(\mathcal{J}, \mathbf{f}) \geq (K-1)2^{K-2} \geq \frac{K-1}{4} \cdot T_{\text{OPT}}(\mathcal{J}, \mathbf{f})$.

Now, suppose the above failure scenario $\mathbf{f}$ happens with probability $Q(\mathbf{f}) > \frac{1}{2}$. Then, based on Equation (5) (see paper), the expected approximation ratio of BATCH-LIST satisfies:

$$\mathbb{E}\left[\frac{T_{\text{BATCH-LIST}}(\mathcal{J})}{T_{\text{OPT}}(\mathcal{J})}\right] > Q(\mathbf{f}) \cdot \frac{T_{\text{BATCH-LIST}}(\mathcal{J}, \mathbf{f})}{T_{\text{OPT}}(\mathcal{J}, \mathbf{f})} > \frac{K-1}{8}.$$

If we fix $K > 8C + 1$, we would get the results if $Q(\mathbf{f}) > \frac{1}{2}$ is true, given any bounded probabilities for the jobs. Intuitively, if a job has a very low failure probability, the probability that it completes successfully in the required batch is also very low. To resolve this issue, we use the following technique: replace each job J_j with a cluster $\mathcal{C}_j$ of n_j jobs that are all identical to J_j, i.e., each with an execution time t_j and a failure probability q_j. We also scale up the number of processors accordingly so that each job can still be executed on a dedicated processor. Then, by choosing n_j wisely, we can make sure that cluster $\mathcal{C}_j$ completes successfully in batch B_{K+j} with high probability, and thus, collectively, the failure scenario $\mathbf{f}$ happens with high probability. In particular, we choose n_j as follows:

$$n_j = \left\lfloor \frac{2^{K+j-1} \ln(1/q_j)}{q_j^{2^{K+j-1}-1}} \right\rfloor \, .$$

Lemma 4. *Under the above choice of n_j and when K is large enough, the probability that any cluster $\mathcal{C}_j$, where $1 \leq j \leq K$, takes exactly $K + j$ batches to complete satisfies:*

$$S_j \geq 1 - 2^K \rho^{2^K} - \rho^{2^{K-1}} \, .$$

Proof. Based on Lemma 3, the probability that cluster $\mathcal{C}_j$ takes exactly $K + j$ batches to complete is given by:

$$S_j = \left(1 - q_j^{2^{K+j}-1}\right)^{n_j} - \left(1 - q_j^{2^{K+j-1}-1}\right)^{n_j} \, . \qquad (3)$$

We now apply the following inequalities that hold for any $x \in [0,1]$ and $n \in \mathbb{N}$:

$$1 - nx \leq (1-x)^n \leq e^{-nx} .$$

In particular, the first inequality comes from the Bernoulli's Inequality, and the second inequality can be derived from the well-known inequality $(1+1/x)^x < e$ for any $x \geq 1$. Applying these two inequalities to Equation (3), we get:

$$\begin{aligned} S_j &\geq \left(1 - q_j^{2^{K+j}}\right)^{n_j} - \left(1 - q_j^{2^{K+j-1}-1}\right)^{n_j} \\ &\geq 1 - n_j q_j^{2^{K+j}} - e^{-n_j q_j^{2^{K+j-1}-1}} . \end{aligned} \tag{4}$$

We will now provide upper bounds for the second term $X_j = n_j q_j^{2^{K+j}}$ and the third term $Y_j = e^{-n_j q_j^{2^{K+j-1}-1}}$.

To bound the second term, we note that $\ln(x) \leq x$ for any $x > 0$ and $n_j \leq \frac{2^{K+j-1}\ln(1/q_j)}{q_j^{2^{K+j-1}-1}}$. The second term then satisfies:

$$\begin{aligned} X_j &\leq \frac{2^{K+j-1}\ln(1/q_j)}{q_j^{2^{K+j-1}-1}} q_j^{2^{K+j}} \\ &\leq \frac{2^{K+j-1}}{q_j^{2^{K+j-1}}} q_j^{2^{K+j}} \\ &= 2^{K+j-1} q_j^{2^{K+j-1}} \\ &\leq 2^{K+j-1} \rho^{2^{K+j-1}} . \end{aligned}$$

Further, we can easily check that $x\rho^x$ is a decreasing function of x when $x\ln(\rho) < -1$. Thus, when K is large enough, and since $j \geq 1$, we have $2^{K+j-1}\rho^{2^{K+j-1}} \leq 2^K \rho^{2^K}$. Therefore, we can get the following upper bound for the second term:

$$X_j \leq 2^K \rho^{2^K} . \tag{5}$$

To bound the third term, we note that $n_j \geq \frac{2^{K+j-1}\ln(1/q_j)}{2 \cdot q_j^{2^{K+j-1}-1}}$, so we can get:

$$\begin{aligned} Y_j &\leq e^{-\frac{2^{K+j-1}\ln(1/q_j)}{2 \cdot q_j^{2^{K+j-1}-1}} q_j^{2^{K+j-1}-1}} \\ &= e^{-2^{K+j-2}\ln(1/q_j)} \\ &= q_j^{2^{K+j-2}} \\ &\leq \rho^{2^{K+j-2}} \\ &\leq \rho^{2^{K-1}} . \end{aligned} \tag{6}$$

The lemma is then proved by substituting Inequalities (5) and (6) into Inequality (4). □

Based on the result of Lemma 4, the probability of the desired failure scenario $\mathbf{f}$ can be computed as:

$$\begin{aligned} Q(\mathbf{f}) &= \prod_{j=1}^{K} S_j \\ &\geq \left(1 - 2^K \rho^{2^K} - \rho^{2^{K-1}}\right)^K \\ &\geq 1 - K2^K \rho^{2^K} - K\rho^{2^{K-1}} . \end{aligned}$$

The last inequality is again due to the Bernoulli's Inequality.

For any constant $\rho < 1$, two terms above, namely, $K2^K\rho^{2^K}$ and $K\rho^{2^{K-1}}$ both tend to 0 as $K \to \infty$. Therefore, there must exist a K^* such that $Q(\mathbf{f}) \geq \frac{1}{2}$. Setting $K = \max(K^*, 8C+1)$ proves the theorem. □

8 Experimental Results for All Speedup Models

Recall that the simulation code for all experiments is publicly available at http://www.github.com/vlefevre/job-scheduling. We present here results for all speedup models.

6.1. Comparison of Algorithms and Priority Rules (for All Speedup Models)

We first compare the performance of different algorithms and study the impact of priority rules on their performance.

Figure 1 shows the normalized makespans for the 11 combinations of algorithms and priority rules under all speedup models. For the MinArea algorithm, priority rules LA and LPT are identical, as the algorithm allocates one processor to all jobs, so only the results of LPT are reported. As we can see, MinArea fares poorly in most cases, because it allocates one processor to each job in order to minimize the area. This results in very long job execution (and re-execution) times, which leads to extremely large makespan. Moreover, allocating only one processor per job also results in idle processors thus resource inefficiency whenever the number of processors is higher than the number of jobs. The MinTime algorithm performs well for the roofline and mix models, but as more overhead is introduced in the communication, Amdahl and power models, it continues to allocate a large number of processors to the jobs in order to minimize the execution time. This leads to a significant increase in the total area and hence degrades the performance. On the other hand, the Lpa and Batch algorithms maintain a good balance between the execution time and area of a job, thus they perform well for all speedup models in terms of both expected performance (bars) and worst-case performance (top endpoints of lines). Independently of the priority rules, Lpa performs the best for the roofline and communication models while Batch performs the best for the other models.

Figure 2 further shows the results of four combinations of P and n with similar performance trends. We notice that these two parameters do have an impact on the performance of Batch under the communication, Amdahl and mix models, in particular at $P = 1000$ and $n = 500$. Indeed, under these models and when P is significantly larger than n, Batch tends to reduce all jobs to similar length and execute them at the same time, which gives the best tradeoff between the area and maximum execution time. In that case, the first batch, where all jobs are executed exactly once, is done almost perfectly. As the makespan of the first batch is dominant under $\lambda = 10^{-7}$, the overall makespan is closer to the lower bound. However, with $P = 1000$ and $n = 500$, there are not enough processors to execute all jobs at the same time. Thus, the performance of Batch becomes worse than that of Lpa.

We also notice that the performance of MinTime under the two mix models becomes better when the number of processors is large compared to the number of jobs (e.g., $P = 10000, n = 100$). Indeed, MinTime is able to simultaneously minimize the execution time of all jobs in this case without using up all the processors, thus achieving near-optimal performance. Note this is not possible with fewer processors, as minimizing the execution time alone

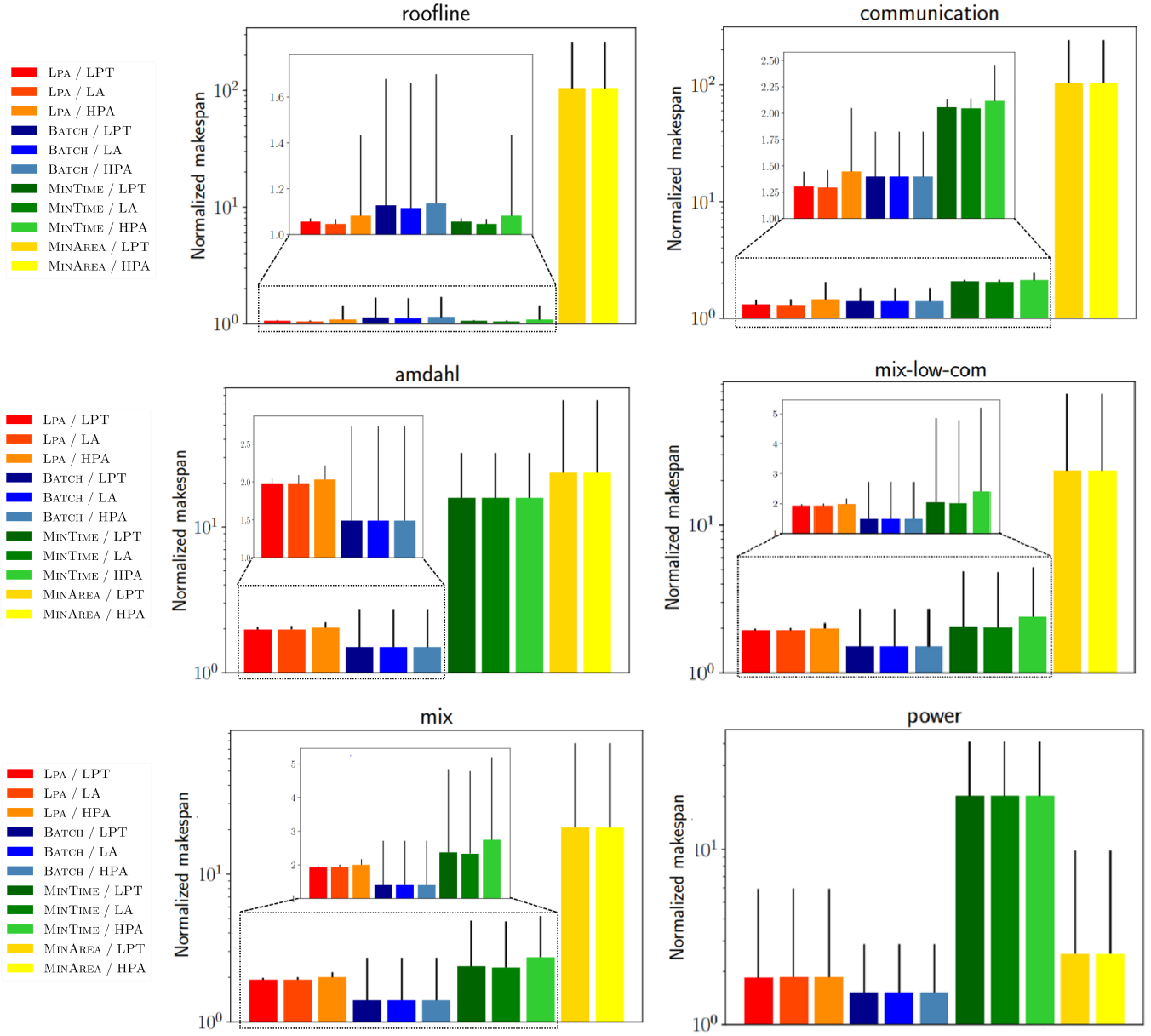

Fig. 1. Performance of different algorithms and priority rules under six speedup models with $P = 7500$, $n = 500$ and $\lambda = 10^{-7}$. The bars represent expected performance and the top endpoints of the lines represent worst-case performance.

for each job will increase the total area, which also plays an important role under such circumstance to have overall good performance.

Comparing the three priority rules, no significant difference is observed. In general, LPT and LA give similar results, and slightly better results than HPA. This is consistent with the results observed in our previous work for scheduling rigid jobs. Given these results, we will only consider the LPT priority rule in the subsequent evaluation. We will also omit the MINAREA and MINTIME algorithms for the models under which they perform badly, while focusing on comparing the expected performance of the remaining algorithms.

6.2. Impact of Different Parameters (for All Speedup Models)

We now study the impact of different parameters on the performance of the algorithms. We start from $P = 7500$, $n = 500$, and $\lambda = 10^{-7}$, and vary one of these parameters in each experiment.

Impact of Number of Processors (P): Figure 3 shows the performance when the number of processors P is varied between 1000 and 15000 for different speedup models. For the roofline model, all three algorithms return the same processor allocation, i.e., the maximum degree of parallelism or the maximum number of processors, for each job. Further, both LPA and MINTIME use the LIST strategy for scheduling, so the two algorithms have exactly the same performance. In contrast, BATCH does not perform as well, because it schedules the jobs in batches, and thus needs to

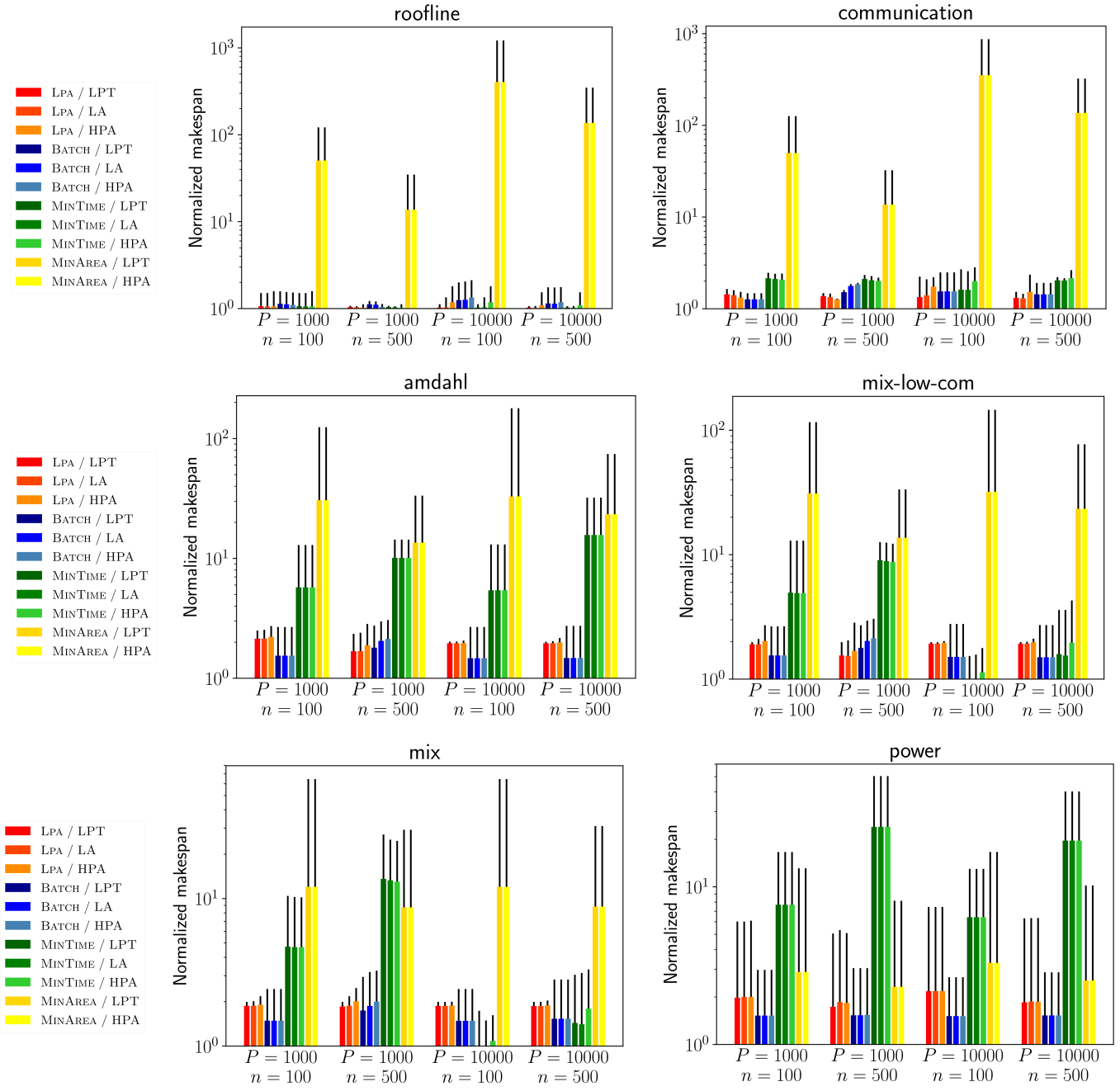

Fig. 2. Performance of different algorithms and priority rules under six speedup models with $\lambda = 10^{-7}$ and four other different combinations of P and n. The bars represent expected performance and the top endpoints of the lines represent worst-case performance.

wait for every job in a batch to finish before starting the next one, which causes delays. The initial up-and-down of the normalized makespans is due to the upper limit (i.e., 4000) we set on the maximum degree of parallelism: when $P \ll 4000$, few processors are wasted so the resulting schedules are very efficient; when $P \gg 4000$, most jobs are fully parallelized and thus completed faster. For BATCH, however, the proportion of idle processors at the end of a batch increases with P, which explains the widening of performance gap from the other two algorithms.

For the communication model, parallelizing a job becomes less efficient due to the extra communication overhead, so BATCH starts to perform better than MINTIME thanks to its smarter processor allocation strategy. Here, both BATCH and LPA have similar processor allocations, so the performance difference between the two algorithms is still induced by the idle times at the end of the batches, which are again increasing with the number of processors.

For the Amdahl's model, the results look very different, as BATCH now outperforms LPA despite the idle time at the end of each batch. This is due to BATCH's ability to better balance the job execution times globally, which becomes more important in this case. Moreover, the trend is not affected by the number of processors.

For the two mix models, LPA and BATCH behave similarly as in the Amdahl's model, because they tend to allocate

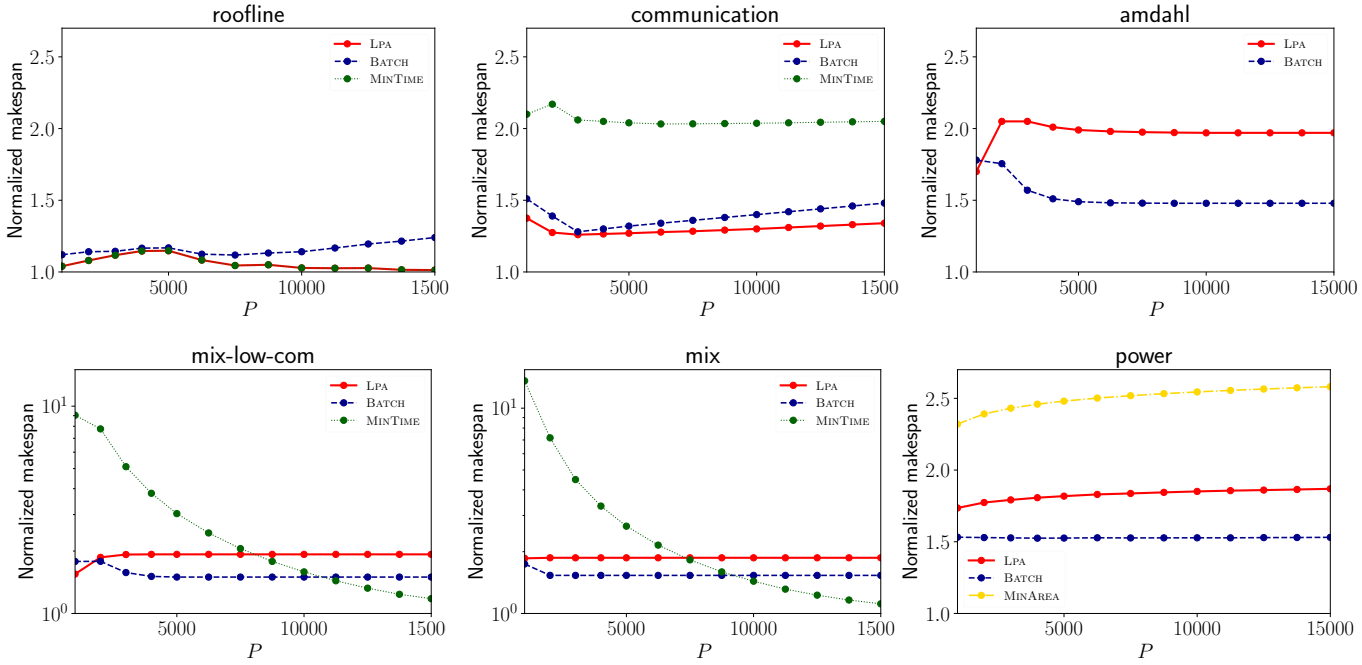

Fig. 3. Performance of the algorithms for different speedup models with $n = 500$, $\lambda = 10^{-7}$ and $P \in [1000, 15000]$.

a relatively small number of processors for each job, thus the maximum degree of parallelism is not reached and the communication cost is relatively small. We also notice that the performance of MINTIME is getting better with increasing number of processors, especially under higher communication cost. Indeed, contrary to the Amdahl's model (where the execution time of a job is minimized when we allocate all the processors), the minimum execution time of a job is achieved with a reasonable number of processors because of the communication overhead. Thus, when P is high enough such that all jobs can be processed in parallel while minimizing their execution times, MINTIME's allocation becomes close to optimal.

Unlike the previous models, the power model has a relatively slow-increasing speedup curve, thus allocating one processor to each job as in MINAREA is not a bad choice. For the same reason, MINTIME that allocates all the processors to a job performs badly, so it is not showed here. The relative performance of LPA and BATCH is similar to that in the Amdahl's and mix models, again due to BATCH's coordinated processor allocation strategy. Because of the jobs' slow speedup curves, the benefit of allocating more processors also gets smaller, thus having more processors barely impacts the performance of the algorithms.

Impact of Number of Jobs (n): Figure 4 shows the performance when the number of jobs n is varied between 100 and 1000. Again, we can see that BATCH performs the worst in the roofline model, gets better than MINTIME in the communication model, and has the best performance in the other models. While the varying number of jobs has a small impact on the performance of LPA, the performance of BATCH improves as the number of jobs increases in the roofline and communication models. Indeed, with a constant number of processors P, having more jobs decreases the number of available processors per job, thus reduces the performance gap between scheduling algorithms due to the idle processors between batches. For the other models, the number of jobs has a small impact even for BATCH. Overall, as the number of jobs increases, the trend in the relative performance of the algorithms is consistent with the previous results we have observed in Figure 3 when the number of processors decreases.

Impact of Error Rate (λ): Figure 5 shows the impact of the error rate λ when it is varied between 10^{-8} (corresponding to 0.03 error per job on average) and 10^{-6} (corresponding to 12 errors per job on average). Once again, the relative performance of the three algorithms remains the same as before under the respective speedup models. While the performance of LPA is barely affected, which is not surprising considering that its processor allocation is performed locally and separately from job scheduling, the performance of BATCH gets worse with increasing error rate λ (and hence the number of failures), which corroborates the theoretical analysis. In particular, when the error rate is small, there are very few failures and almost all jobs will complete in one batch. In this case, the processor allocation procedure of BATCH is very precise. With increased error rate, more failures will occur and thus more batches will be introduced, causing scheduling inefficiencies from both idle times between the batches and possible imprecision in the processor allocations (especially with a large batch, since the actual number of failures may deviate significantly from the anticipated values). Finally, although the processor allocation is also performed locally for MINTIME and MINAREA, the effect of increasing λ is similar to that of increasing P (or the opposite to that of increasing n): when there are more failures, we spend more time processing few large jobs that fail a lot, meaning that after some time only very few jobs are not finished yet. This effectively increases the total number of processors for these jobs or reduces the total number of jobs.

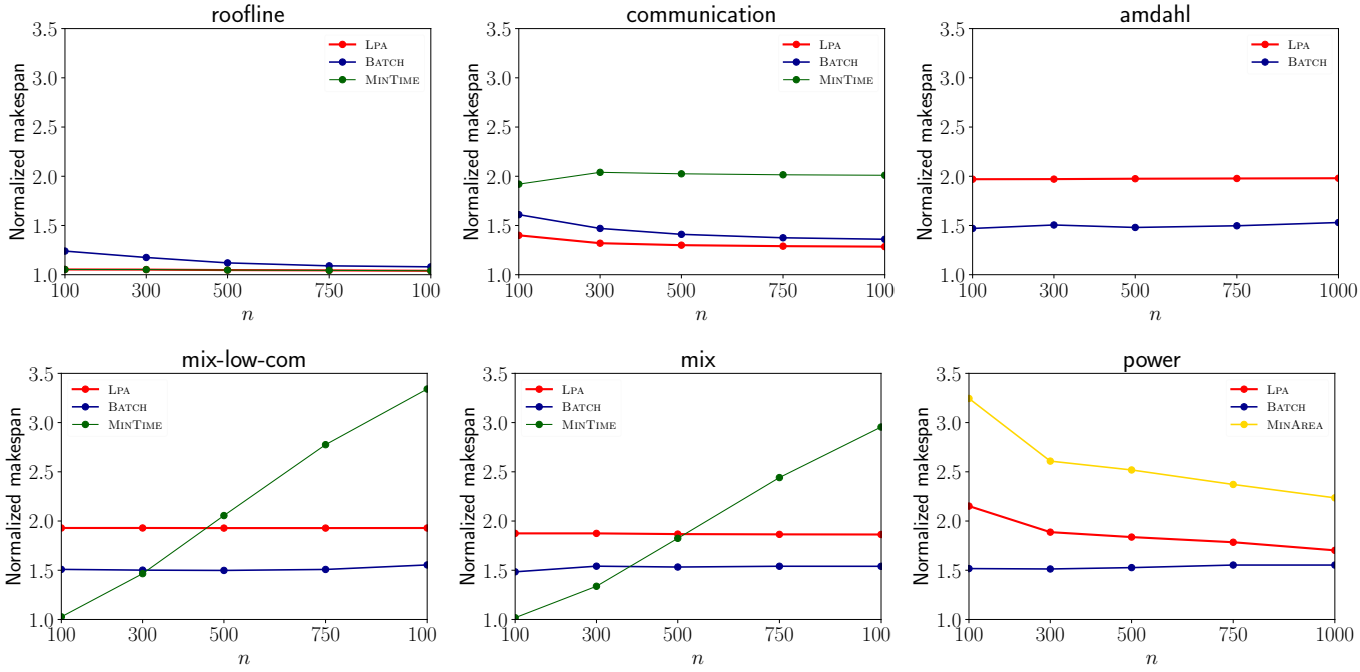

Fig. 4. Performance of the algorithms for different speedup models with $P = 7500$, $\lambda = 10^{-7}$ and $n \in [100, 1000]$.

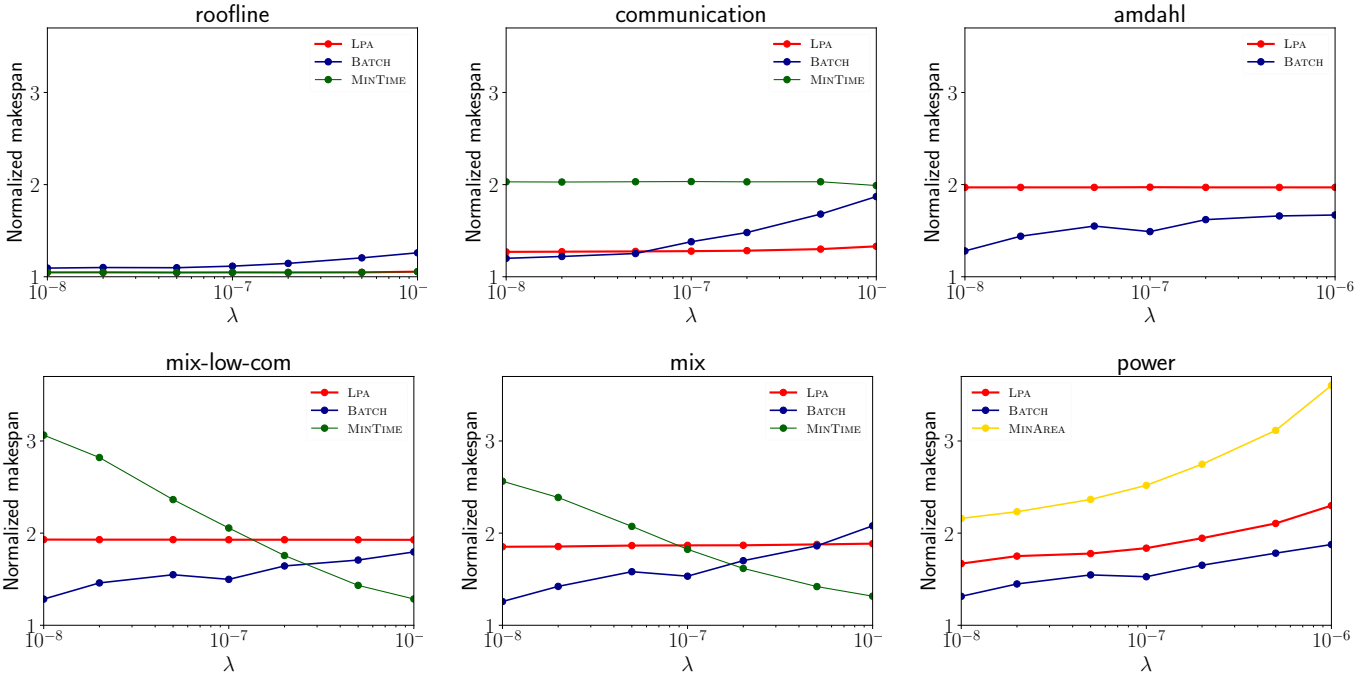

Fig. 5. Performance of the algorithms for different speedup models with $P = 7500$, $n = 500$ and $\lambda \in [10^{-8}, 10^{-6}]$.