Longest Processing Time rule for identical parallel machines scheduling revisited

Federico Della Croce[1,2] Rosario Scatamacchia [1]

[1]DIGEP - Politecnico di Torino, Torino, Italy

[2]CNR, IEIIT, Torino, Italy

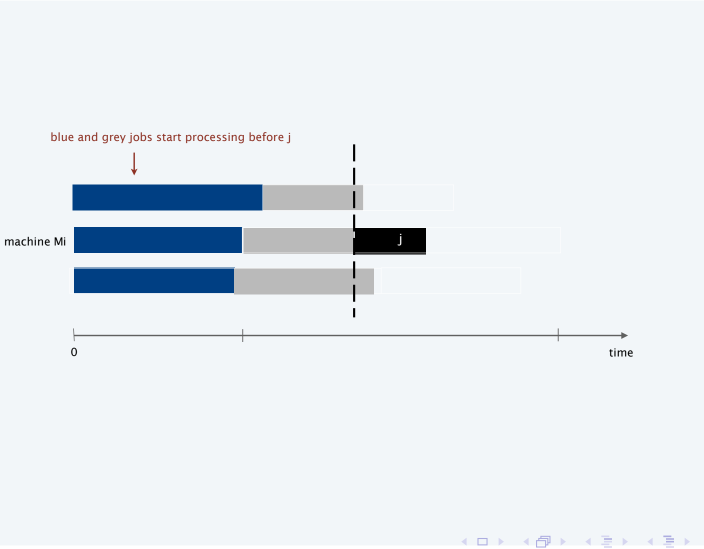
GOThA et Bermudes - Tours, 26-27 septembre 2017

Outline

1. Introduction
2. *LPT* rule
3. *LPT* revisited
4. Improving the *LPT* bound
5. From approximation to heuristics
6. Computational testing

Introduction

- We consider problem $P_m||C_{max}$ where the goal is to schedule n jobs on m identical parallel machines M_i $(i = 1, \ldots, m)$ minimizing the makespan.



Introduction

- We consider problem $P_m||C_{max}$ where the goal is to schedule n jobs on m identical parallel machines M_i $(i = 1, \ldots, m)$ minimizing the makespan.

- For this NP-hard problem, we revisit the famous Longest Processing Time (LPT) rule proposed by Graham - 1969.

Introduction

- We consider problem $P_m||C_{max}$ where the goal is to schedule n jobs on m identical parallel machines M_i $(i = 1, \ldots, m)$ minimizing the makespan.

- For this NP-hard problem, we revisit the famous Longest Processing Time (LPT) rule proposed by Graham - 1969.

- We employ Linear Programming to analyze the worst case performance of a simple modification of LPT that manages to improve the longstanding Graham's bound (from $\frac{4}{3} - \frac{1}{3m}$ to $\frac{4}{3} - \frac{1}{3(m-1)}$) for $m \geq 3$.

Introduction

- We consider problem $P_m||C_{max}$ where the goal is to schedule n jobs on m identical parallel machines M_i $(i = 1, \ldots, m)$ minimizing the makespan.
- For this NP-hard problem, we revisit the famous Longest Processing Time (LPT) rule proposed by Graham - 1969.
- We employ Linear Programming to analyze the worst case performance of a simple modification of LPT that manages to improve the longstanding Graham's bound (from $\frac{4}{3} - \frac{1}{3m}$ to $\frac{4}{3} - \frac{1}{3(m-1)}$) for $m \geq 3$.
- Then, we move from approximation to heuristics. By generalizing the proposed approach we obtain a simple $O(n \log n)$ procedure that drastically improves upon the performances of the LPT rule.

Introduction

- We consider problem $P_m||C_{max}$ where the goal is to schedule n jobs on m identical parallel machines M_i $(i = 1, \ldots, m)$ minimizing the makespan.
- For this NP-hard problem, we revisit the famous Longest Processing Time (LPT) rule proposed by Graham - 1969.
- We employ Linear Programming to analyze the worst case performance of a simple modification of LPT that manages to improve the longstanding Graham's bound (from $\frac{4}{3} - \frac{1}{3m}$ to $\frac{4}{3} - \frac{1}{3(m-1)}$) for $m \geq 3$.
- Then, we move from approximation to heuristics. By generalizing the proposed approach we obtain a simple $O(n \log n)$ procedure that drastically improves upon the performances of the LPT rule.
- On 780 benchmark literature instances (Iori and Martello 2008), the new procedure wins 513 times, ties 224 times and loses 43 times against LPT.

Introduction

- *LPT* rule: sort the jobs $1, ..., n$ in non-ascending order of their processing times p_j $(j = 1, \ldots, n)$ and then assign one job at a time to the machine whose load is smallest so far.

Introduction

- *LPT* rule: sort the jobs $1, ..., n$ in non-ascending order of their processing times p_j $(j = 1, \ldots, n)$ and then assign one job at a time to the machine whose load is smallest so far.

- Assume the jobs sorted by non-increasing p_j $(p_j \geq p_{j+1},\ j = 1, \ldots, n-1)$.

Introduction

- *LPT* rule: sort the jobs $1, ..., n$ in non-ascending order of their processing times p_j $(j = 1, \ldots, n)$ and then assign one job at a time to the machine whose load is smallest so far.

- Assume the jobs sorted by non-increasing p_j $(p_j \geq p_{j+1},\ j = 1, \ldots, n-1)$.

- Denote the solution values of the *LPT* schedule and the optimal makespan by C_m^{LPT} and C_m^* respectively, where index m indicates the number of machines.

Introduction

- *LPT* rule: sort the jobs $1, ..., n$ in non-ascending order of their processing times p_j $(j = 1, \ldots, n)$ and then assign one job at a time to the machine whose load is smallest so far.
- Assume the jobs sorted by non-increasing p_j $(p_j \geq p_{j+1},\ j = 1, \ldots, n-1)$.
- Denote the solution values of the *LPT* schedule and the optimal makespan by C_m^{LPT} and C_m^* respectively, where index m indicates the number of machines.
- Denote by $r_k = \frac{C_m^{LPT}}{C_m^*}$ the approximation ratio of the *LPT* schedule with k jobs assigned to the machine yielding the maximum completion time (the critical machine)

$P_m||C_{max}$ problem and LPT rule properties

- $C_m^* \geq p_1$.

$P_m||C_{max}$ problem and LPT rule properties

- $C_m^* \geq p_1$.
- $C_m^* \geq \frac{\sum_{j=1}^{n} p_j}{m}$.

$P_m||C_{max}$ problem and LPT rule properties

- $C_m^* \geq p_1$.
- $C_m^* \geq \frac{\sum\limits_{j=1}^{n} p_j}{m}$.
- $C_m^{LPT} = C_m^*$ if $p_{j'} > \frac{C_m^*}{3}$ (j' denotes the critical job).

$P_m||C_{max}$ problem and LPT rule properties

- $C_m^* \geq p_1$.
- $C_m^* \geq \frac{\sum\limits_{j=1}^{n} p_j}{m}$.
- $C_m^{LPT} = C_m^*$ if $p_{j'} > \frac{C_m^*}{3}$ (j' denotes the critical job).
- $C_m^{LPT} \leq \frac{\sum\limits_{j=1}^{j'} p_j}{m} + p_{j'}(1 - \frac{1}{m}) \leq C_m^* + p_{j'}(1 - \frac{1}{m})$ - [Graham 1969].

$P_m||C_{max}$ problem and LPT rule properties

- $C_m^* \geq p_1$.
- $C_m^* \geq \frac{\sum\limits_{j=1}^{n} p_j}{m}$.
- $C_m^{LPT} = C_m^*$ if $p_{j'} > \frac{C_m^*}{3}$ (j' denotes the critical job).
- $C_m^{LPT} \leq \frac{\sum\limits_{j=1}^{j'} p_j}{m} + p_{j'}(1 - \frac{1}{m}) \leq C_m^* + p_{j'}(1 - \frac{1}{m})$ - [Graham 1969].
- For each job i assigned by LPT in position j on a machine:

$$p_i \leq \frac{C_m^*}{j} \text{ - [Chen 1993].}$$

LPT rule properties:

Known *LPT* approximation ratios.

- $r_1 = 1$.

LPT rule properties:

Known *LPT* approximation ratios.

- $r_1 = 1$.
- $r_2 = \frac{4}{3} - \frac{1}{3(m-1)}$ - [Chen 1993].

LPT rule properties:

Known *LPT* approximation ratios.

- $r_1 = 1$.
- $r_2 = \frac{4}{3} - \frac{1}{3(m-1)}$ - [Chen 1993].
- $r_k = \frac{k+1}{k} - \frac{1}{km} \quad k \geq 3$ [Coffman and Sethi 1976 - Graham 1969 for $k = 3$].

LPT rule properties:

Known LPT approximation ratios.

- $r_1 = 1$.
- $r_2 = \frac{4}{3} - \frac{1}{3(m-1)}$ - [Chen 1993].
- $r_k = \frac{k+1}{k} - \frac{1}{km} \quad k \geq 3$ [Coffman and Sethi 1976 - Graham 1969 for $k = 3$].

Notice that

- $r_2 = 1$ for $m = 2$;
- $r_2 = r_4$ for $m = 3$;
- $r_2 < r_4$ for $m \geq 4$;
- $r_k < r_{k+1}$ for $k \geq 3$

LPT rule properties:

Known LPT approximation ratios.

- $r_1 = 1$.
- $r_2 = \frac{4}{3} - \frac{1}{3(m-1)}$ - [Chen 1993].
- $r_k = \frac{k+1}{k} - \frac{1}{km} \quad k \geq 3$ [Coffman and Sethi 1976 - Graham 1969 for $k = 3$].

Notice that

- $r_2 = 1$ for $m = 2$;
- $r_2 = r_4$ for $m = 3$;
- $r_2 < r_4$ for $m \geq 4$;
- $r_k < r_{k+1}$ for $k \geq 3$

$\Longrightarrow$ **Improving r_3 improves LPT.**
We concentrate then on instances where the critical job is in position 3.

Tight worst-case examples for *LPT*

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.

Tight worst-case examples for *LPT*

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.

- $C^*_{m=2} = 6, \qquad C^{LPT}_{m=2} = 7, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{7}{6}.$

Tight worst-case examples for LPT

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.
- $C^*_{m=2} = 6, \qquad C^{LPT}_{m=2} = 7, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{7}{6}.$
- 3 machines, 7 jobs with jobs $1, 2$ of length 5, jobs $3, 4$ of length 4 and jobs $5, 6, 7$ of length 3.

Tight worst-case examples for LPT

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.

- $C^*_{m=2} = 6, \qquad C^{LPT}_{m=2} = 7, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{7}{6}.$

- 3 machines, 7 jobs with jobs $1, 2$ of length 5, jobs $3, 4$ of length 4 and jobs $5, 6, 7$ of length 3.

- $C^*_{m=3} = 9, \qquad C^{LPT}_{m=3} = 11, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{11}{9}.$

Tight worst-case examples for LPT

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.

- $C^*_{m=2} = 6, \qquad C^{LPT}_{m=2} = 7, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{7}{6}.$

- 3 machines, 7 jobs with jobs $1, 2$ of length 5, jobs $3, 4$ of length 4 and jobs $5, 6, 7$ of length 3.

- $C^*_{m=3} = 9, \qquad C^{LPT}_{m=3} = 11, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{11}{9}.$

- m machines, $2m + 1$ jobs with jobs $1, 2$ of length $2m - 1$, jobs $3, 4$ of length $2m - 2$ ... jobs $2m - 1, 2m, 2m + 1$ of length m.

Tight worst-case examples for LPT

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.
- $C^*_{m=2} = 6, \qquad C^{LPT}_{m=2} = 7, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{7}{6}.$
- 3 machines, 7 jobs with jobs $1, 2$ of length 5, jobs $3, 4$ of length 4 and jobs $5, 6, 7$ of length 3.
- $C^*_{m=3} = 9, \qquad C^{LPT}_{m=3} = 11, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{11}{9}.$
- m machines, $2m + 1$ jobs with jobs $1, 2$ of length $2m - 1$, jobs $3, 4$ of length $2m - 2$ ... jobs $2m - 1, 2m, 2m + 1$ of length m.
- $C^*_{m} = 3m, \qquad C^{LPT}_{m} = 4m - 1, \qquad r_3 = \frac{4m-1}{3m} = \frac{4}{3} - \frac{1}{3m}.$

Tight worst-case examples for LPT

- 2 machines - 5 jobs with jobs $1, 2$ of length 3 and jobs $3, 4, 5$ of length 2.
- $C^*_{m=2} = 6, \qquad C^{LPT}_{m=2} = 7, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{7}{6}$.
- 3 machines, 7 jobs with jobs $1, 2$ of length 5, jobs $3, 4$ of length 4 and jobs $5, 6, 7$ of length 3.
- $C^*_{m=3} = 9, \qquad C^{LPT}_{m=3} = 11, \qquad r_3 = \frac{4}{3} - \frac{1}{3m} = \frac{11}{9}$.
- m machines, $2m + 1$ jobs with jobs $1, 2$ of length $2m - 1$, jobs $3, 4$ of length $2m - 2$ ... jobs $2m - 1, 2m, 2m + 1$ of length m.
- $C^*_m = 3m, \qquad C^{LPT}_m = 4m - 1, \qquad r_3 = \frac{4m-1}{3m} = \frac{4}{3} - \frac{1}{3m}$.
- Worst-case always occurs with $2m + 1 = n$ jobs where the critical job is job $2m + 1 = n$ in position 3 and when $C^*_m = \sum_{i=1}^{n} p_i/m$.

LPT revisited

- We assume that the LPT critical job is the last one, namely $j' = n$. Otherwise, we would have further jobs after the critical job which do not affect the makespan provided by LPT but can contribute to increase the optimal solution value.

LPT revisited

- We assume that the LPT critical job is the last one, namely $j' = n$. Otherwise, we would have further jobs after the critical job which do not affect the makespan provided by LPT but can contribute to increase the optimal solution value.

- We analyze for $m \geq 3$:
  - $2m + 2 \leq j' = n \leq 3m$ (or else the critical job would be in position ≥ 4);

LPT revisited

- We assume that the *LPT* critical job is the last one, namely $j' = n$. Otherwise, we would have further jobs after the critical job which do not affect the makespan provided by *LPT* but can contribute to increase the optimal solution value.

- We analyze for $m \geq 3$:
    - $2m + 2 \leq j' = n \leq 3m$ (or else the critical job would be in position ≥ 4);
    - $j' = n = 2m + 1$.

LPT revisited

- We assume that the *LPT* critical job is the last one, namely $j' = n$. Otherwise, we would have further jobs after the critical job which do not affect the makespan provided by *LPT* but can contribute to increase the optimal solution value.

- We analyze for $m \geq 3$:
  - $2m + 2 \leq j' = n \leq 3m$ (or else the critical job would be in position ≥ 4);
  - $j' = n = 2m + 1$.

- We employ **Linear Programming** to perform the analysis.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.
- We consider an LP model where we arbitrarily set the value C_m^{LPT} to 1 and minimize the value of C_m^*.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.
- We consider an LP model where we arbitrarily set the value C_m^{LPT} to 1 and minimize the value of C_m^*.
- L denotes the starting time of job n, i.e. $C_m^{LPT} = L + p_n$.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.
- We consider an LP model where we arbitrarily set the value C_m^{LPT} to 1 and minimize the value of C_m^*.
- L denotes the starting time of job n, i.e. $C_m^{LPT} = L + p_n$.
- C_1 is the compl. time of the non-crit. machine processing at least 3 jobs.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.
- We consider an LP model where we arbitrarily set the value C_m^{LPT} to 1 and minimize the value of C_m^*.
- L denotes the starting time of job n, i.e. $C_m^{LPT} = L + p_n$.
- C_1 is the compl. time of the non-crit. machine processing at least 3 jobs.
- C_2 is the sum of compl. times of the other $(m-2)$ machines, i.e.

  $$C_2 = \sum_{j=1}^{n} p_j - C_1 - (L + p_n).$$

LPT revisited: $2m+2 \le n \le 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\le \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \ge 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.
- We consider an LP model where we arbitrarily set the value C_m^{LPT} to 1 and minimize the value of C_m^*.
- L denotes the starting time of job n, i.e. $C_m^{LPT} = L + p_n$.
- C_1 is the compl. time of the non-crit. machine processing at least 3 jobs.
- C_2 is the sum of compl. times of the other $(m-2)$ machines, i.e. $C_2 = \sum_{j=1}^{n} p_j - C_1 - (L + p_n)$.
- Due to list scheduling, condition $\frac{C_2}{m-2} \ge L$ holds.

LPT revisited: $2m + 2 \leq n \leq 3m$

Proposition

If LPT schedules at least 3 jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 5$.

Sketch of proof.

- We assume n in position 3, or else either r_2 holds or at least r_4 holds. Hence, LPT schedules at least another job in position ≥ 3.
- We consider an LP model where we arbitrarily set the value C_m^{LPT} to 1 and minimize the value of C_m^*.
- L denotes the starting time of job n, i.e. $C_m^{LPT} = L + p_n$.
- C_1 is the compl. time of the non-crit. machine processing at least 3 jobs.
- C_2 is the sum of compl. times of the other $(m-2)$ machines, i.e.
  $$C_2 = \sum_{j=1}^{n} p_j - C_1 - (L + p_n).$$
- Due to list scheduling, condition $\frac{C_2}{m-2} \geq L$ holds.
- As n is in position 3, condition $p_n \leq \frac{C_m^*}{3}$ holds.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- We associate non-negative variables π and ξ with p_n and $\sum_{j=1}^{n} p_j$.
- We associate non-negative variables c_1, c_2, l, opt with C_1, C_2, L and C_m^*.

LPT revisited: $2m+2 \le n \le 3m$ (LP formulation)

- We associate non-negative variables π and ξ with p_n and $\sum_{j=1}^{n} p_j$.
- We associate non-negative variables c_1, c_2, l, opt with C_1, C_2, L and C_m^*.
- The following LP model is implied:

$$
\begin{aligned}
\text{minimize} \quad & opt && (1)\\
\text{subject to} \quad & -m \cdot opt + \xi \le 0 && (2)\\
& 3 \cdot \pi - c_1 \le 0 && (3)\\
& l - c_1 \le 0 && (4)\\
& (m-2)l - c_2 \le 0 && (5)\\
& c_1 + l + \pi + c_2 - \xi = 0 && (6)\\
& l + \pi = 1 && (7)\\
& \pi - \frac{opt}{3} \le 0 && (8)\\
& \pi, \xi, c_1, c_2, l, opt \ge 0 && (9)
\end{aligned}
$$

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.

- Constraint (2) represents the bound $C^*_m \geq \frac{\sum_{j=1}^{n} p_j}{m}$.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.
- Constraint (2) represents the bound $C^*_m \geq \frac{\sum_{j=1}^{n} p_j}{m}$.
- Constraint (3) states that the value of c_1 is at the least $3p_n$, since 3 jobs with processing time $\geq p_n$ are assigned to a non critical machine.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.
- Constraint (2) represents the bound $C_m^* \geq \frac{\sum_{j=1}^{n} p_j}{m}$.
- Constraint (3) states that the value of c_1 is at the least $3p_n$, since 3 jobs with processing time $\geq p_n$ are assigned to a non critical machine.
- Constraint (4) states that the processing time of the critical machine before the last job is loaded is less than the completion time of the other machine processing at least three jobs.

LPT revisited: $2m+2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.
- Constraint (2) represents the bound $C_m^* \geq \frac{\sum_{j=1}^{n} p_j}{m}$.
- Constraint (3) states that the value of c_1 is at the least $3p_n$, since 3 jobs with processing time $\geq p_n$ are assigned to a non critical machine.
- Constraint (4) states that the processing time of the critical machine before the last job is loaded is less than the completion time of the other machine processing at least three jobs.
- Constraint (5) fulfills the list scheduling requirement.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.
- Constraint (2) represents the bound $C^*_m \geq \frac{\sum_{j=1}^{n} p_j}{m}$.
- Constraint (3) states that the value of c_1 is at the least $3p_n$, since 3 jobs with processing time $\geq p_n$ are assigned to a non critical machine.
- Constraint (4) states that the processing time of the critical machine before the last job is loaded is less than the completion time of the other machine processing at least three jobs.
- Constraint (5) fulfills the list scheduling requirement.
- Constraint (6) guarantees that variable ξ represents $\sum_{j=1}^{n} p_j$

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The minimization of the objective function (1), after setting w.l.o.g LPT solution value to 1 (constraint (7)), provides an upper bound on the performance ratio of LPT rule.
- Constraint (2) represents the bound $C_m^* \geq \frac{\sum_{j=1}^{n} p_j}{m}$.
- Constraint (3) states that the value of c_1 is at the least $3p_n$, since 3 jobs with processing time $\geq p_n$ are assigned to a non critical machine.
- Constraint (4) states that the processing time of the critical machine before the last job is loaded is less than the completion time of the other machine processing at least three jobs.
- Constraint (5) fulfills the list scheduling requirement.
- Constraint (6) guarantees that variable ξ represents $\sum_{j=1}^{n} p_j$
- Constraint (8) represents condition $p_n \leq \frac{C_m^*}{3}$.
- Constraints (9) state that all variables are non-negative.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The proposed LP model is continuous and contains just 6 variables and 7 constraints for any fixed m.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The proposed LP model is continuous and contains just 6 variables and 7 constraints for any fixed m.
- By strong duality (and a little bit of reverse engineering) it is possible to show that in the optimal solution, for any $m \geq 5$, the variables values are as follows

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The proposed LP model is continuous and contains just 6 variables and 7 constraints for any fixed m.
- By strong duality (and a little bit of reverse engineering) it is possible to show that in the optimal solution, for any $m \geq 5$, the variables values are as follows

$$\pi = \frac{m-1}{4m-5}; \qquad \xi = \frac{3m(m-1)}{4m-5};$$

$$c_1 = \frac{3(m-1)}{4m-5}; \qquad c_2 = \frac{(m-2)(3(m-1)-1)}{4m-5};$$

$$l = \frac{3(m-1)-1}{4m-5}; \qquad opt = \frac{3(m-1)}{4m-5}.$$

LPT revisited: $2m+2 \leq n \leq 3m$ (LP formulation)

- The proposed LP model is continuous and contains just 6 variables and 7 constraints for any fixed m.
- By strong duality (and a little bit of reverse engineering) it is possible to show that in the optimal solution, for any $m \geq 5$, the variables values are as follows

$$\pi = \frac{m-1}{4m-5}; \qquad \xi = \frac{3m(m-1)}{4m-5};$$
$$c_1 = \frac{3(m-1)}{4m-5}; \qquad c_2 = \frac{(m-2)(3(m-1)-1)}{4m-5};$$
$$l = \frac{3(m-1)-1}{4m-5}; \qquad opt = \frac{3(m-1)}{4m-5}.$$

- Correspondingly, for any $m \geq 5$, we have $\frac{C_m^{LPT}}{C_m^*} \leq 1/opt = \frac{4m-5}{3(m-1)} = \frac{4}{3} - \frac{1}{3(m-1)}$.

LPT revisited: $2m + 2 \leq n \leq 3m$ (LP formulation)

- The proposed LP model is continuous and contains just 6 variables and 7 constraints for any fixed m.
- By strong duality (and a little bit of reverse engineering) it is possible to show that in the optimal solution, for any $m \geq 5$, the variables values are as follows

$$\pi = \frac{m-1}{4m-5}; \qquad \xi = \frac{3m(m-1)}{4m-5};$$

$$c_1 = \frac{3(m-1)}{4m-5}; \qquad c_2 = \frac{(m-2)(3(m-1)-1)}{4m-5};$$

$$l = \frac{3(m-1)-1}{4m-5}; \qquad opt = \frac{3(m-1)}{4m-5}.$$

- Correspondingly, for any $m \geq 5$, we have $\frac{C_m^{LPT}}{C_m^*} \leq 1/opt = \frac{4m-5}{3(m-1)} = \frac{4}{3} - \frac{1}{3(m-1)}$.
- Notice that **this bound is not tight**.

LPT revisited: $2m+2 \leq n \leq 3m$

- A more general results, provided below, actually holds.

Proposition

If LPT schedules at least k jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{k+1}{k} - \frac{1}{k(m-1)}$ for $m \geq k+2$.

LPT revisited: $2m + 2 \leq n \leq 3m$

- A more general results, provided below, actually holds.

Proposition

If LPT schedules at least k jobs on a non crit. machine before assigning the crit. job, then LPT has an approx. bound $\leq \frac{k+1}{k} - \frac{1}{k(m-1)}$ *for* $m \geq k + 2$.

- For $m \leq 4$, by lp-modeling and partial enumeration it is possible to obtain the following result.

Proposition

In $P_m||C_{max}$ *instances with* $2m + 2 \leq n \leq 3m$, *LPT (with job n critical) has an approximation ratio* $\leq \frac{4}{3} - \frac{1}{3(m-1)}$ *for* $3 \leq m \leq 4$.

LPT revisited: further subcases

The following propositions also hold

Proposition

In $P_m||C_{max}$ instances with $n \leq 2m$ and $m \geq 3$, LPT has an approximation ratio $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.

LPT revisited: further subcases

The following propositions also hold

Proposition

In $P_m||C_{max}$ *instances with* $n \leq 2m$ *and* $m \geq 3$*,* LPT *has an approximation ratio* $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.

Proposition

In $P_m||C_{max}$, $m \geq 3$ *and instances with* $n = 2m+1$*, if* LPT *loads at least three jobs on a machine before the critical job, then it has an approximation ratio* $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.

LPT revisited: further subcases

The following propositions also hold

Proposition

In $P_m||C_{max}$ instances with $n \leq 2m$ and $m \geq 3$, LPT has an approximation ratio $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.

Proposition

In $P_m||C_{max}$, $m \geq 3$ and instances with $n = 2m + 1$, if LPT loads at least three jobs on a machine before the critical job, then it has an approximation ratio $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.

- The only case remaining is then related to instances with $n = 2m + 1$ where LPT schedules job n only in third position and n is critical.

Improving LPT for $n = 2m + 1$

- We consider a slight algorithmic variation where a set of the sorted jobs is first loaded on a machine and then LPT is applied on the remaining job set.

Improving LPT for $n = 2m + 1$

- We consider a slight algorithmic variation where a set of the sorted jobs is first loaded on a machine and then LPT is applied on the remaining job set.
- We denote this variant as $LPT(\mathcal{S})$ where $\mathcal{S}$ represents the set of jobs assigned all together to a machine first.

Improving LPT for $n = 2m + 1$

- We consider a slight algorithmic variation where a set of the sorted jobs is first loaded on a machine and then LPT is applied on the remaining job set.
- We denote this variant as $LPT(\mathcal{S})$ where $\mathcal{S}$ represents the set of jobs assigned all together to a machine first.

We consider the following Algorithm 1.

Input: $P_m||C_{max}$ instance with n jobs and $m \geq 3$ machines.
- Apply LPT yielding a schedule with makespan z_1 and $k-1$ jobs on the critical machine before job n.
- Apply $LPT' = LPT(\{n\})$ with solution value z_2.
- Apply $LPT'' = LPT(\{(n-k+1), ..., n\})$ with solution value z_3.
- Return $\min\{z_1, z_2, z_3\}$.

In practice, this algorithm applies LPT first and then re-applies LPT *after* having loaded on a machine first either its critical job n alone or the tuple of k jobs $n-k+1, ..., n$.

Handling instances with n jobs and $j' = 2m + 1 \neq n$

We consider first the case where $j' \neq n$ and there are jobs processed after the critical job in LPT and one of such jobs is critical in either LPT' or LPT''.

Handling instances with n jobs and $j' = 2m + 1 \neq n$

We consider first the case where $j' \neq n$ and there are jobs processed after the critical job in LPT and one of such jobs is critical in either LPT' or LPT''.

Proposition

In $P_m||C_{max}$ instances where there are jobs processed after the critical job in the LPT solution and one of such jobs (say job l) is critical in either LPT' or LPT'' schedules, Algorithm 1 has a performance guarantee of $\frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)}$.

Handling instances with n jobs and $j' = 2m + 1 \neq n$

We consider first the case where $j' \neq n$ and there are jobs processed after the critical job in LPT and one of such jobs is critical in either LPT' or LPT''.

Proposition

In $P_m||C_{max}$ instances where there are jobs processed after the critical job in the LPT solution and one of such jobs (say job l) is critical in either LPT' or LPT'' schedules, Algorithm 1 has a performance guarantee of $\frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)}$.

Proof hints (formal proof needs some more algebra):

- it is sufficient to exploit the difference between $\sum_{j=1}^{j'} p_j$ and $\sum_{j=1}^{n} p_j$.

Handling instances with n jobs and $j' = 2m + 1 \neq n$

We consider first the case where $j' \neq n$ and there are jobs processed after the critical job in LPT and one of such jobs is critical in either LPT' or LPT''.

Proposition

In $P_m||C_{max}$ instances where there are jobs processed after the critical job in the LPT solution and one of such jobs (say job l) is critical in either LPT' or LPT'' schedules, Algorithm 1 has a performance guarantee of $\frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)}$.

Proof hints (formal proof needs some more algebra):

- it is sufficient to exploit the difference between $\sum_{j=1}^{j'} p_j$ and $\sum_{j=1}^{n} p_j$.
- If $\sum_{j=j'+1}^{n} p_j$ is large enough, then $\frac{\sum_{j=1}^{j'} p_j}{m} + p_{j'}/m \ll \frac{\sum_{j=1}^{n} p_j}{m} + p_l/m$, namely, the bound on the LPT approx. ratio becomes small enough;

Handling instances with n jobs and $j' = 2m + 1 \neq n$

We consider first the case where $j' \neq n$ and there are jobs processed after the critical job in LPT and one of such jobs is critical in either LPT' or LPT''.

Proposition

In $P_m||C_{max}$ instances where there are jobs processed after the critical job in the LPT solution and one of such jobs (say job l) is critical in either LPT' or LPT'' schedules, Algorithm 1 has a performance guarantee of $\frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)}$.

Proof hints (formal proof needs some more algebra):

- it is sufficient to exploit the difference between $\sum_{j=1}^{j'} p_j$ and $\sum_{j=1}^{n} p_j$.
- If $\sum_{j=j'+1}^{n} p_j$ is large enough, then $\frac{\sum_{j=1}^{j'} p_j}{m} + p_{j'}/m \ll \frac{\sum_{j=1}^{n} p_j}{m} + p_l/m$, namely, the bound on the LPT approx. ratio becomes small enough;
- if $\sum_{j=j'+1}^{n} p_j$ is small enough, then the approx. ratio of LPT' or LPT'' also becomes small enough.

Handling instances with $n = 2m + 1$ jobs and $j' = n = 2m + 1$

- Note that LPT must couple jobs $1, \ldots, m$ respectively with jobs $2m, \ldots, m + 1$ on the m machines before scheduling job $2m + 1$, or else LPT has an approximation ratio $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.

Handling instances with $n = 2m + 1$ jobs and $j' = n = 2m + 1$

- Note that LPT must couple jobs $1, \ldots, m$ respectively with jobs $2m, \ldots, m + 1$ on the m machines before scheduling job $2m + 1$, or else LPT has an approximation ratio $\leq \left(\frac{4}{3} - \frac{1}{3(m-1)}\right)$.
- Hence, the LPT schedule is as follows

$$
\begin{aligned}
&M_1 : p_1, p_{2m} \\
&M_2 : p_2, p_{2m-1} \\
&\ldots \\
&M_{m-1} : p_{m-1}, p_{m+2} \\
&M_m : p_m, p_{m+1}
\end{aligned}
$$

where job $2m + 1$ will be assigned to the machine with the least processing time.

Handling instances with $n = 2m + 1$ jobs and $j' = n = 2m + 1$

We consider two specific cases:

1. $p_{2m+1} \geq p_1 - p_m$. $\Longrightarrow$ The LPT' schedule is as follows

$$
\begin{aligned}
&M_1 : p_{2m+1}, p_m, p_{2m} \\
&M_2 : p_1, p_{2m-1} \\
&M_3 : p_2, p_{2m-2} \\
&\ldots \\
&M_{m-1} : p_{m-2}, p_{m+2} \\
&M_m : p_{m-1}, p_{m+1}
\end{aligned}
$$

Handling instances with $n = 2m + 1$ jobs and $j' = n = 2m + 1$

We consider two specific cases:

1. $p_{2m+1} \geq p_1 - p_m$. $\Longrightarrow$ The LPT' schedule is as follows

$$
\begin{aligned}
&M_1 : p_{2m+1}, p_m, p_{2m} \\
&M_2 : p_1, p_{2m-1} \\
&M_3 : p_2, p_{2m-2} \\
&\dots \\
&M_{m-1} : p_{m-2}, p_{m+2} \\
&M_m : p_{m-1}, p_{m+1}
\end{aligned}
$$

   with subcases

   1. The LPT' makespan is on M_1.
   2. The LPT' makespan is on M_2,...M_m.

Handling instances with $n = 2m+1$ jobs and $j' = n = 2m+1$

We consider two specific cases:

1. $p_{2m+1} \geq p_1 - p_m$. $\Longrightarrow$ The LPT' schedule is as follows

$$
\begin{aligned}
&M_1 : p_{2m+1}, p_m, p_{2m} \\
&M_2 : p_1, p_{2m-1} \\
&M_3 : p_2, p_{2m-2} \\
&\ldots \\
&M_{m-1} : p_{m-2}, p_{m+2} \\
&M_m : p_{m-1}, p_{m+1}
\end{aligned}
$$

   with subcases
   1. The LPT' makespan is on M_1.
   2. The LPT' makespan is on M_2,...M_m.

2. $p_{2m+1} < p_1 - p_m$.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- If LPT' is not optimal, then $C^*_m \geq p_{m-1} + p_m$.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- If LPT' is not optimal, then $C^*_m \geq p_{m-1} + p_m$.
- We get the following result.

Proposition

If $p_{2m+1} \geq p_1 - p_m$ and LPT' makespan is equal to $p_{2m+1} + p_m + p_{2m}$, then the proposed algorithm has an approximation ratio not superior to $\frac{7}{6}$.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- If LPT' is not optimal, then $C^*_m \geq p_{m-1} + p_m$.
- We get the following result.

Proposition

If $p_{2m+1} \geq p_1 - p_m$ and LPT' makespan is equal to $p_{2m+1} + p_m + p_{2m}$, then the proposed algorithm has an approximation ratio not superior to $\frac{7}{6}$.

- Proof: we again employ Linear Programming to evaluate the performance of LPT'. We consider non-negative variables x_j associated with p_j $(j = 1, \ldots, n)$ and a positive parameter $OPT > 0$ associated with C^*_m.

Case $j' = n = 2m+1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

The LP model.

$$
\begin{aligned}
\text{maximize} \quad & x_{(2m+1)} + x_m + x_{2m} && (10)\\
\text{subject to} \quad & x_{(m-1)} + x_m \leq OPT && (11)\\
& x_{(2m-1)} + x_{2m} + x_{(2m+1)} \leq OPT && (12)\\
& x_{(2m+1)} - (x_1 - x_m) \geq 0 && (13)\\
& x_1 - x_{(m-1)} \geq 0 && (14)\\
& x_{(m-1)} - x_m \geq 0 && (15)\\
& x_m - x_{(m+1)} \geq 0 && (16)\\
& x_{(m+1)} - x_{(2m-1)} \geq 0 && (17)\\
& x_{(2m-1)} - x_{2m} \geq 0 && (18)\\
& x_{2m} - x_{(2m+1)} \geq 0 && (19)\\
& x_1, x_{(m-1)}, x_m, x_{(m+1)}, x_{(2m-1)}, x_{2m}, x_{(2m+1)} \geq 0 && (20)
\end{aligned}
$$

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- The objective function value (10) represents an upper bound on the worst case performance of the algorithm.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- The objective function value (10) represents an upper bound on the worst case performance of the algorithm.
- Constraints (11)–(12) correspond to $C^*_m \geq p_{m-1} + p_m$ and $C^*_m \geq p_{2m-1} + p_{2m} + p_{2m+1}$.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- The objective function value (10) represents an upper bound on the worst case performance of the algorithm.
- Constraints (11)–(12) correspond to $C_m^* \geq p_{m-1} + p_m$ and $C_m^* \geq p_{2m-1} + p_{2m} + p_{2m+1}$.
- Constraint (13) simply represents the initial assumption $p_{2m+1} \geq p_1 - p_m$.

Case $j' = n = 2m+1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- The objective function value (10) represents an upper bound on the worst case performance of the algorithm.
- Constraints (11)–(12) correspond to $C^*_m \geq p_{m-1} + p_m$ and $C^*_m \geq p_{2m-1} + p_{2m} + p_{2m+1}$.
- Constraint (13) simply represents the initial assumption $p_{2m+1} \geq p_1 - p_m$.
- Constraints (14)–(19) state that the considered relevant jobs are sorted by non-increasing processing times.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- The objective function value (10) represents an upper bound on the worst case performance of the algorithm.
- Constraints (11)–(12) correspond to $C_m^* \geq p_{m-1} + p_m$ and $C_m^* \geq p_{2m-1} + p_{2m} + p_{2m+1}$.
- Constraint (13) simply represents the initial assumption $p_{2m+1} \geq p_1 - p_m$.
- Constraints (14)–(19) state that the considered relevant jobs are sorted by non-increasing processing times.
- Constraints (20) indicate that the variables are non-negative.

Case $j' = n = 2m + 1$, $p_{2m+1} \geq p_1 - p_m$, LPT' makespan is on M_1

- The objective function value (10) represents an upper bound on the worst case performance of the algorithm.
- Constraints (11)–(12) correspond to $C^*_m \geq p_{m-1} + p_m$ and $C^*_m \geq p_{2m-1} + p_{2m} + p_{2m+1}$.
- Constraint (13) simply represents the initial assumption $p_{2m+1} \geq p_1 - p_m$.
- Constraints (14)–(19) state that the considered relevant jobs are sorted by non-increasing processing times.
- Constraints (20) indicate that the variables are non-negative.
- Further viable constraints where not necessary to reach the required result. By setting $OPT = 1$, the cost function has value $\frac{7}{6}$.

Further cases and subcases for $j' = n = 2m + 1$

By means of further LP models, the following results hold

Further cases and subcases for $j' = n = 2m + 1$

By means of further LP models, the following results hold

Proposition

If $p_{2m+1} \geq p_1 - p_m$ *and* LPT' *makespan is on* $M_2, ..., M_m$*, then* LPT' *has a performance guarantee of* $\frac{15}{13}$ *for* $m = 3$ *and* $\frac{4}{3} - \frac{1}{2m-1}$ *for* $m \geq 4$*.*

Further cases and subcases for $j' = n = 2m + 1$

By means of further LP models, the following results hold

Proposition

If $p_{2m+1} \geq p_1 - p_m$ and LPT' makespan is on $M_2, \ldots, M_m$, then LPT' has a performance guarantee of $\frac{15}{13}$ for $m = 3$ and $\frac{4}{3} - \frac{1}{2m-1}$ for $m \geq 4$.

Proposition

If $p_{2m+1} < p_1 - p_m$, LPT has a performance guarantee not superior to $\frac{15}{13}$ for $m = 3$ and $\frac{4}{3} - \frac{1}{2m-1}$ for $m \geq 4$.

Further cases and subcases for $j' = n = 2m + 1$

By means of further LP models, the following results hold

Proposition

If $p_{2m+1} \geq p_1 - p_m$ and LPT' makespan is on $M_2, ..., M_m$, then LPT' has a performance guarantee of $\frac{15}{13}$ for $m = 3$ and $\frac{4}{3} - \frac{1}{2m-1}$ for $m \geq 4$.

Proposition

If $p_{2m+1} < p_1 - p_m$, LPT has a performance guarantee not superior to $\frac{15}{13}$ for $m = 3$ and $\frac{4}{3} - \frac{1}{2m-1}$ for $m \geq 4$.

Putting things together, the following Theorem holds

Theorem

The proposed algorithm has an approximation ratio not superior to $\frac{4}{3} - \frac{1}{3(m-1)}$ for $m \geq 3$.

LPT, LPT' and LPT'' w.r.t. $m = 2$

- For $m = 2$, $\frac{4}{3} - \frac{1}{3(m-1)} = 1$, hence a different analysis is required.

LPT, LPT' and LPT'' w.r.t. $m = 2$

- For $m = 2$, $\frac{4}{3} - \frac{1}{3(m-1)} = 1$, hence a different analysis is required.
- We know that for $m = 2$ and $j' = n = 4$, LPT is optimal and that for $j' = n \geq 7$ the approx. ratio of LPT is not superior to $9/8$ [Coffman and Sethi 1976].

LPT, LPT' and LPT'' w.r.t. $m = 2$

- For $m = 2$, $\frac{4}{3} - \frac{1}{3(m-1)} = 1$, hence a different analysis is required.
- We know that for $m = 2$ and $j' = n = 4$, LPT is optimal and that for $j' = n \geq 7$ the approx. ratio of LPT is not superior to $9/8$ [Coffman and Sethi 1976].
- We managed to prove that for $m = 2$ and $j' = n = 6$, the approx. ratio of LPT is not superior to $9/8$.

LPT, LPT' and LPT'' w.r.t. $m = 2$

- For $m = 2$, $\frac{4}{3} - \frac{1}{3(m-1)} = 1$, hence a different analysis is required.
- We know that for $m = 2$ and $j' = n = 4$, LPT is optimal and that for $j' = n \geq 7$ the approx. ratio of LPT is not superior to $9/8$ [Coffman and Sethi 1976].
- We managed to prove that for $m = 2$ and $j' = n = 6$, the approx. ratio of LPT is not superior to $9/8$.
- We managed to prove that for $m = 2$ and $j' = n = 5$, the best sol among the ones reached by LPT, LPT' and LPT'' is optimal.

LPT, LPT' and LPT'' w.r.t. $m = 2$

- For $m = 2$, $\frac{4}{3} - \frac{1}{3(m-1)} = 1$, hence a different analysis is required.

- We know that for $m = 2$ and $j' = n = 4$, LPT is optimal and that for $j' = n \geq 7$ the approx. ratio of LPT is not superior to $9/8$ [Coffman and Sethi 1976].

- We managed to prove that for $m = 2$ and $j' = n = 6$, the approx. ratio of LPT is not superior to $9/8$.

- We managed to prove that for $m = 2$ and $j' = n = 5$, the best sol among the ones reached by LPT, LPT' and LPT'' is optimal.

- The case for $m = 2$ where there are jobs processed after the critical job in the LPT solution and one of such jobs (say job l) is critical in either LPT' or LPT'' schedules keeps the same performance guarantee of $\frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)} = \frac{4}{3} - \frac{10}{39} = 14/13 < 9/8$.

LPT, LPT' and LPT'' w.r.t. $m = 2$

- For $m = 2$, $\frac{4}{3} - \frac{1}{3(m-1)} = 1$, hence a different analysis is required.

- We know that for $m = 2$ and $j' = n = 4$, LPT is optimal and that for $j' = n \geq 7$ the approx. ratio of LPT is not superior to $9/8$ [Coffman and Sethi 1976].

- We managed to prove that for $m = 2$ and $j' = n = 6$, the approx. ratio of LPT is not superior to $9/8$.

- We managed to prove that for $m = 2$ and $j' = n = 5$, the best sol among the ones reached by LPT, LPT' and LPT'' is optimal.

- The case for $m = 2$ where there are jobs processed after the critical job in the LPT solution and one of such jobs (say job l) is critical in either LPT' or LPT'' schedules keeps the same performance guarantee of $\frac{4}{3} - \frac{7m-4}{3(3m^2+m-1)} = \frac{4}{3} - \frac{10}{39} = 14/13 < 9/8$.

- Putting things together, for $m = 2$, the approx. ratio of the proposed algorithm is not superior to $9/8$.

From approximation to heuristics

- W.r.t. worst-case analysis, we remarked that LPT' was necessary to improve Graham's bound for $m \geq 3$, while LPT'' was necessary for $m \geq 2$.

From approximation to heuristics

- W.r.t. worst-case analysis, we remarked that LPT' was necessary to improve Graham's bound for $m \geq 3$, while LPT'' was necessary for $m \geq 2$.

- Remarkably, for $m \geq 3$, the relevant subcase was the one with $p_{2m+1} \geq p_1 - p_m$ and LPT' required to schedule p_{2m+1} first and then apply list scheduling first the sorted jobset $p_1, ..., p_m$ according to LPT and then to the sorted jobset $p_{m+1}, ..., p_{2m}$ always according to LPT.

From approximation to heuristics

- W.r.t. worst-case analysis, we remarked that LPT' was necessary to improve Graham's bound for $m \geq 3$, while LPT'' was necessary for $m \geq 2$.
- Remarkably, for $m \geq 3$, the relevant subcase was the one with $p_{2m+1} \geq p_1 - p_m$ and LPT' required to schedule p_{2m+1} first and then apply list scheduling first the sorted jobset $p_1, ..., p_m$ according to LPT and then to the sorted jobset $p_{m+1}, ..., p_{2m}$ always according to LPT.
- We propose then an alternative approach that splits the sorted job set in tuples of m consecutive jobs $(1, \ldots, m; m+1, \ldots, 2m;$ etc.$)$ and sorts the tuples in non-increasing order of the difference between the largest job and the smallest job in the tuple. Then a list scheduling is applied to the set of sorted tuples. We denote this approach as $SLACK$.

From approximation to heuristics

The $SLACK$ heuristic:

Input: $P_m||C_{max}$ instance m machines and n jobs with processing times p_j $(j = 1, \ldots, n)$.
- Sort items by non-increasing p_j.
- Consider $\left\lceil \frac{n}{m} \right\rceil$ tuples of size m given by jobs $1, \ldots, m; m+1, \ldots, 2m$, etc..
If n is not multiple of m, add dummy jobs with null proc. time in the last tuple.
- For each tuple, compute the associated slack, namely
$p_1 - p_m, p_{(m+1)} - p_{2m}, \ldots, p_{(n-m+1)} - p_n$.
- Sort tuples by non-increasing slack and then fill a list of consecutive jobs in the sorted tuples.
- Apply List Scheduling to this job ordering and return the solution.

From approximation to heuristics

The $SLACK$ heuristic:

Input: $P_m||C_{max}$ instance m machines and n jobs with processing times p_j $(j = 1, \ldots, n)$.
- Sort items by non-increasing p_j.
- Consider $\left\lceil \frac{n}{m} \right\rceil$ tuples of size m given by jobs $1, \ldots, m; m+1, \ldots, 2m$, etc..
If n is not multiple of m, add dummy jobs with null proc. time in the last tuple.
- For each tuple, compute the associated slack, namely
$p_1 - p_m, p_{(m+1)} - p_{2m}, \ldots, p_{(n-m+1)} - p_n$.
- Sort tuples by non-increasing slack and then fill a list of consecutive jobs in the sorted tuples.
- Apply List Scheduling to this job ordering and return the solution.

Since the construction and sorting of the tuples can be performed in $\mathcal{O}(m \log m)$, the running time of $SLACK$ is $\mathcal{O}(n \log n)$ due to the initial jobs LPT sorting.

Computational testing

We compared $SLACK$ to LPT on benchmark literature instances (Iori, Martello 2008)

Computational testing

We compared *SLACK* to *LPT* on benchmark literature instances (Iori, Martello 2008)

- Two classical classes of instances from literature are considered: *uniform instances* (França et al. 1994) and *non-uniform instances* (Frangioni et al. 2004).

Computational testing

We compared $SLACK$ to LPT on benchmark literature instances (Iori, Martello 2008)

- Two classical classes of instances from literature are considered: *uniform instances* (França et al. 1994) and *non-uniform instances* (Frangioni et al. 2004).
- In *uniform instances* the processing times are integer uniformly distributed in the range $[a, b]$. In *non-uniform instances*, 98% of the processing times are integer uniformly distributed in $[0.9(b-a), b]$ while the remaining ones are uniformly distributed in $[a, 0.2(b-a)]$. For both classes, we have $a = 1; b = 100, 1000, 10000$.

Computational testing

We compared $SLACK$ to LPT on benchmark literature instances (Iori, Martello 2008)

- Two classical classes of instances from literature are considered: *uniform instances* (França et al. 1994) and *non-uniform instances* (Frangioni et al. 2004).
- In *uniform instances* the processing times are integer uniformly distributed in the range $[a, b]$. In *non-uniform instances*, 98% of the processing times are integer uniformly distributed in $[0.9(b-a), b]$ while the remaining ones are uniformly distributed in $[a, 0.2(b-a)]$. For both classes, we have $a = 1; b = 100, 1000, 10000$.
- For each class, the following values were considered for the number of machines and jobs: $m = 5, 10, 25$ and $n = 10, 50, 100, 500, 1000$.

Computational testing

We compared *SLACK* to *LPT* on benchmark literature instances (Iori, Martello 2008)

- Two classical classes of instances from literature are considered: *uniform instances* (França et al. 1994) and *non-uniform instances* (Frangioni et al. 2004).
- In *uniform instances* the processing times are integer uniformly distributed in the range $[a, b]$. In *non-uniform instances*, 98% of the processing times are integer uniformly distributed in $[0.9(b-a), b]$ while the remaining ones are uniformly distributed in $[a, 0.2(b-a)]$. For both classes, we have $a = 1; b = 100, 1000, 10000$.
- For each class, the following values were considered for the number of machines and jobs: $m = 5, 10, 25$ and $n = 10, 50, 100, 500, 1000$.
- For each pair (m, n) with $m < n$, 10 instances were generated for a total of 780 instances.

			SLACK wins		draws		*LPT* wins	
$[a, b]$	m	Instances	#	(%)	#	(%)	#	(%)
1-100	5	50	31	(62.0)	16	(32.0)	3	(6.0)
	10	40	32	(80.0)	8	(20.0)	0	(0.0)
	25	40	23	(57.5)	17	(42.5)	0	(0.0)
1-1000	5	50	39	(78.0)	10	(20.0)	1	(2.0)
	10	40	40	(100.0)	0	(0.0)	0	(0.0)
	25	40	27	(67.5)	12	(30.0)	1	(2.5)
1-10000	5	50	39	(78.0)	10	(20.0)	1	(2.0)
	10	40	40	(100.0)	0	(0.0)	0	(0.0)
	25	40	28	(70.0)	10	(25.0)	2	(5.0)

Table: $P_m||C_{max}$ non uniform instances.

Computational testing

			SLACK wins		draws		*LPT* wins	
$[a, b]$	m	Instances	#	(%)	#	(%)	#	(%)
1-100	5	50	12	(24.0)	37	(74.0)	1	(2.0)
	10	40	14	(35.0)	20	(50.0)	6	(15.0)
	25	40	10	(25.0)	29	(72.5)	1	(2.5)
1-1000	5	50	32	(64.0)	15	(30.0)	3	(6.0)
	10	40	27	(67.5)	5	(12.5)	8	(20.0)
	25	40	24	(60.0)	12	(30.0)	4	(10.0)
1-10000	5	50	36	(72.0)	12	(24.0)	2	(4.0)
	10	40	37	(92.5)	0	(0.0)	3	(7.5)
	25	40	22	(55.0)	11	(27.5)	7	(17.5)

Table: $P_m||C_{max}$ uniform instances.

Computational testing

- $SLACK$ shows up to be clearly superior to LPT: on 780 benchmark literature instances, $SLACK$ wins 513 times, ties 224 times and loses 43 times only.

Computational testing

- $SLACK$ shows up to be clearly superior to LPT: on 780 benchmark literature instances, $SLACK$ wins 513 times, ties 224 times and loses 43 times only.
- If LPT'' is added to SLACK, then $SLACK+LPT''$ compared to LPT wins 529 times, ties 213 times and loses 38 times only.