On Solving Two-processor Scheduling Problem with Energy Constraint

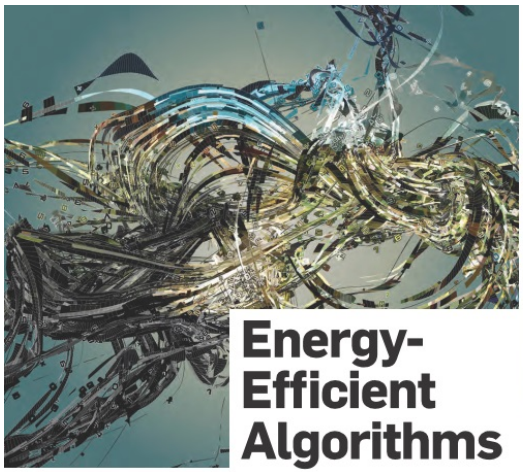
Yu. Zakharova

Sobolev Institute of Mathematics SB RAS

The reported study was funded by Russian Science Foundation, project number 22-71-10015.

Motivation (Parallel and Multiprocessor Jobs)

- Parallel jobs require more than one processor at the same time.
- Some jobs can not be performed asynchronously on modern computers. Such situation takes place in multiprocessor graphics cards, where the memory capacity of one processor is not sufficient.
- Many computer systems offer some kinds of parallelism. The energy efficient scheduling of parallel jobs arises in testing and reliable computing, parallel applications on graphics cards, computer control systems and others.



Multiprocessor and Parallel Jobs

Parallel Jobs

- **Rigid jobs**: the number of required processors is given and fixed ($size_j$).
- **Moldable jobs**: the number of required processors is chosen by the scheduler before starting a job, and is not changed until the job termination (δ_j).
- **Malleable jobs**: the number of required processors is chosen by the scheduler, and can be changed at runtime (δ_j).

Multiprocessor Jobs

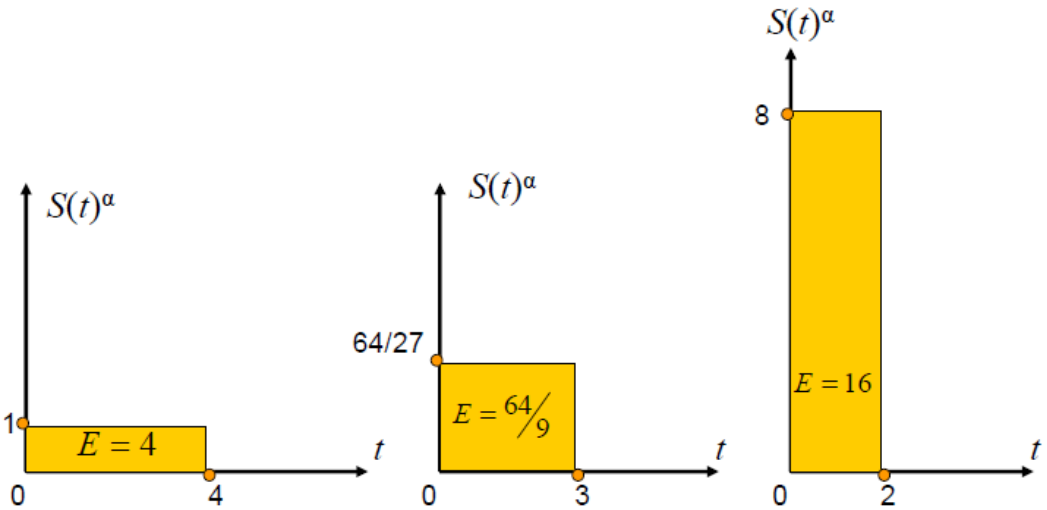
- **Single mode jobs**: the set of required processors is given and fixed (fix_j).
- **Multimode jobs**: alternative sets of processors may be used (set_j).

Uniform and non-uniform partition of work between processors.

Speed Scaling Scheduling

Modern microprocessors ($\alpha = 1.11$ for Intel PXA 270, $\alpha = 1.62$ for Pentium M770, $\alpha = 1.66$ for a TCP offload engine, $\alpha = 3$ for CMOS devices) can run at variable speed.
High speeds result in higher performance but also high energy consumption. Lower speeds save energy but performance degrades.

$$Energy = \int_{t_0}^{t_1} s^{\alpha}(t)\mathrm{d}t$$



Speed Scaling Scheduling

Processors and Jobs

$m = 2$ speed-scalable processors

$\mathcal{J} = \{1, \ldots, n\}$ is the set of jobs:
V_j is the processing volume (work) of job j
$W_j := \frac{V_j}{m_j}$ is the work on one processor

E is the energy budget

Parameters

Preemption and migration are characterized for the systems with single image of the memory.
Non-preemptive instances arise in systems with distributed memory.

Homogeneous Model in Speed-scaling

If a processor runs at speed s then the energy consumption is s^{α} units of energy per time unit, where $\alpha > 1$ is a constant (practical studies show that $\alpha \leq 3$).

It is supposed that a continuous spectrum of processor speeds is available.

E is the energy budget.

The aim is to find a feasible schedule with minimum makespan (total completion time) so that the energy consumption is not greater than a given energy budget.

Previous Research: Classic

Makespan

Drozdowski (2009): poly for rigid jobs, pmtn, r_j
approx for rigid jobs, r_j
Brucker (2020), Du, Leung (1989): rigid jobs: NP-hard, strongly NP-hard for prec

Total Completion Time

Lee and Cai (1999): rigid jobs: strongly NP-hard
Schwiegelshohn et. al. (1998), J. Turek et. al. (1994): approximation algorithms for rigid jobs
Hoogeveen (1994): single-mode jobs: NP-hard
Cai (1998): 2-approximation algorithm for single-mode jobs

Previous Research: Energy

Makespan

Pruhs, van Stee (2007), Bunde (2009): poly for single processor, r_j
approx for multiple processors, r_j
Bampis et.al. (2014): approx for prec, r_j

Total Completion Time

Pruhs et. al. (2008), Bunde (2009): poly for single processor
Shabtay, Kaspi (2006): approx for multiple processors

Report Structure

- Problem Statement
- Previous Research
- Parallel and Single Mode Jobs
- Makespan and Total Completion Time
- Conclusion and Further Research

Rigid Jobs: $P2|size_j, energy|C_{\max}$

Lemma

Problem $P2|size_j, energy|C_{\max}$ is NP-hard.

Two-Stage Algorithm

At the first stage, we find a lower bound on the objective and calculate processing times of jobs using auxiliary convex programs, KKT conditions, and the Ellipsoid method. Then, at the second stage, we transform our problem to the classic scheduling problem without speed scaling and use list-type scheduling via greedy rule to obtain feasible solutions.

Lower Bound

$$T \to \min, \tag{1}$$

$$\max_{j \in \mathcal{J}} p_j \leq T, \tag{2}$$

$$\frac{1}{2} \sum_{j \in \mathcal{J}} size_j p_j \leq T, \tag{3}$$

$$\sum_{j \in \mathcal{J}} size_j W_j^{\alpha} p_j^{1-\alpha} \leq E, \tag{4}$$

$$p_j \geq 0, \ j \in \mathcal{J}. \tag{5}$$

Lower Bound

Inequality (3)

$$\max_{j \in \mathcal{J}} W_j \leq \frac{1}{2} \sum_{j \in \mathcal{J}} size_j W_j. \tag{6}$$

Inequality (2)

$$\sum_{j \in \mathcal{J}} size_j \leq 2, \tag{7}$$

Both inequalities

$$\max_{j=2,\ldots,n} W_j \leq \frac{1}{(2 - size_1)} \sum_{j=2}^{n} size_j W_j. \tag{8}$$

Approximation Algorithm

Non-preemptive List-scheduling

Whenever a subset of processors falls idle, the algorithm assigns a rigid job that does not require more processors than are available.

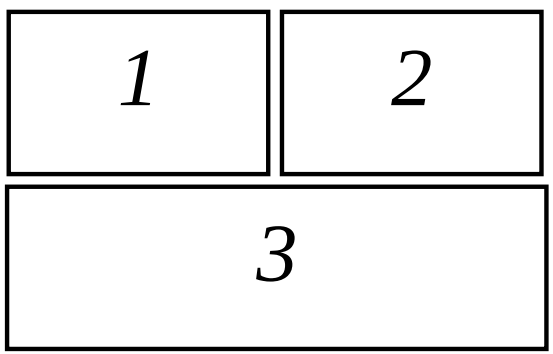
Theorem

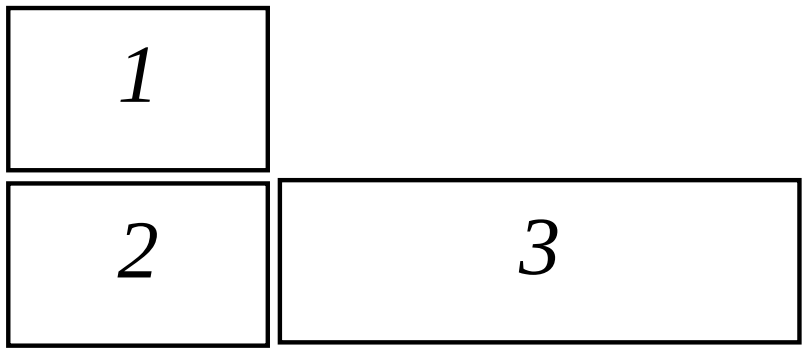
A $\frac{3}{2}$-approximate schedule can be found in polynomial time for $P2|size_j, energy|C_{\max}$.
A $\frac{5}{2}$-approximate schedule can be found in polynomial time for $P2|r_j, size_j, energy|C_{\max}$.

Approximation Guarantee

$W_1=W_2=1$, $W_3=2$, $size_1=size_2=size_3=1$

$C_{max}(OPT)=2$, $C_{max}(LS)=3$





Convex Program

$$T \to \min, \tag{9}$$

$$\frac{1}{2}\sum_{j\in\mathcal{J}} p_j size_j \le T, \tag{10}$$

$$C_j \le T,\ j \in \mathcal{J}, \tag{11}$$

$$p_j \le C_j,\ j \in \mathcal{J}, \tag{12}$$

$$C_j + p_{j'} \le C_{j'},\ (j, j') \in A, \tag{13}$$

$$\sum_{j\in\mathcal{J}} size_j W_j^{\alpha} p_j^{1-\alpha} \le E, \tag{14}$$

$$C_j \ge 0,\ p_j \ge 0,\ j \in \mathcal{J}. \tag{15}$$

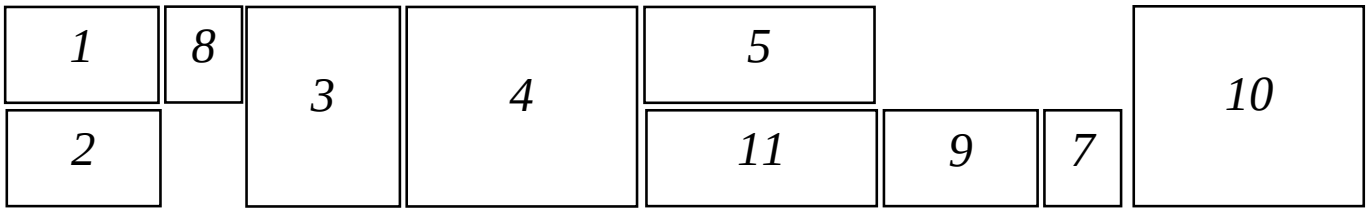
2-approximate schedule

We construct a feasible schedule by the "precedence-dependent list-scheduling" algorithm.

Approximation Algorithm

$C_{max}(LS)=T_1+T_2$

$C_{max}(OPT)>=max\{T_1,(T_1+T_2)/2\}$



$T_1(1)$ $T_1(8)$ T_2 T_2 T_2 $T_1(9)$ T_2 $T_1(10)$

Notations and Variables

$\mathcal{J}_1$ is the set of single-processor jobs.
$\mathcal{J}_{12}$ is the set of 2-processors jobs.
$\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_{12}$.

Intervals $I_i = [s_i, s_i + l_i)$, $i = 1, \ldots, \gamma \leq n$.
The length of the last interval l_γ is unknown in advance and considered here as variable.
Variable $p_{ji} \geq 0$ is the processing time of job j in interval I_i.

Convex Program

$$l_\gamma \to \min, \tag{16}$$

$$\frac{1}{2}\sum_{j\in\mathcal{J}_1} p_{ji} \le l_i - \sum_{j\in\mathcal{J}_{12}} p_{ji},\ i=1,\dots,\gamma, \tag{17}$$

$$\max_{j\in\mathcal{J}_1}\{p_{ji}\} \le l_i - \sum_{j\in\mathcal{J}_{12}} p_{ji},\ i=1,\dots,\gamma, \tag{18}$$

$$\sum_{j\in\mathcal{J}_1}\left(\sum_{i=1}^{\gamma} p_{ji}\right)^{1-\alpha} W_j^\alpha + \sum_{j\in\mathcal{J}_{12}} 2\left(\sum_{i=1}^{\gamma} p_{ji}\right)^{1-\alpha} W_j^\alpha \le E, \tag{19}$$

$$p_{ji} = 0,\ i=1,\dots,\gamma,\ j\in\{j'\in\mathcal{J}:\ r_{j'} > s_i\}. \tag{20}$$

Exact schedule

For each interval, we construct a feasible schedule, initially placing 2-processors jobs and using McNaughton's algorithm for single-processor jobs.

Convex Program

$$l_\gamma \to \min, \tag{21}$$

$$\sum_{j\in\mathcal{J}_1} p_{ji} \le l_i - \sum_{j\in\mathcal{J}_{12}} p_{ji},\ i = 1,\dots,\gamma, \tag{22}$$

$$\sum_{j\in\mathcal{J}_2} p_{ji} \le l_i - \sum_{j\in\mathcal{J}_{12}} p_{ji},\ i = 1,\dots,\gamma, \tag{23}$$

$$\sum_{j\in\mathcal{J}_1\cup\mathcal{J}_2} \left(\sum_{i=1}^{\gamma} p_{ji}\right)^{1-\alpha} W_j^\alpha + \sum_{j\in\mathcal{J}_{12}} 2\left(\sum_{i=1}^{\gamma} p_{ji}\right)^{1-\alpha} W_j^\alpha \le E, \tag{24}$$

$$p_{ji} = 0,\ i = 1,\dots,\gamma,\ j \in \{j' \in \mathcal{J} :\ r_{j'} > s_i\}. \tag{25}$$

Exact schedule

For each interval, we construct a feasible schedule, initially placing 2-processors jobs and then single-processor jobs.

Convex Program

$$p_1 + p_{12} \to \min, \tag{26}$$

$$p_1 + p_{12} = p_2 + p_{12}, \tag{27}$$

$$2p_{12}\left(\frac{W_{12}}{p_{12}}\right)^{\alpha} + p_1\left(\frac{W_1}{p_1}\right)^{\alpha} + p_2\left(\frac{W_2}{p_2}\right)^{\alpha} = E. \tag{28}$$

Notations

$W_1 = \sum_{j \in \mathcal{J}_1} W_j$
$W_2 = \sum_{j \in \mathcal{J}_2} W_j$
$W_{12} = \sum_{j \in \mathcal{J}_{12}} W_j$

Rigid Jobs: $P2|size_j, energy|\sum C_j$

Single-Processor Problem

Given an instance $P2$, we generate instance $P1$:
$W'_j = \frac{W_j}{2}$ for $size_j = 1$, $W'_j = W_j$ for $size_j = 2$ and $E' = \frac{E}{2}$.

Reindex jobs in non-decreasing of volumes W'_j, and find optimal durations p'_j.

Two-processor Problem

Calculate processing times of jobs for $(P2)$: $p_j = 2p'_j$ for single-processor jobs and $p_j = p'_j$ for two-processor jobs.
Assign job j to the first available processor if j requires one processor or to the two processors when both of them are available if j is a two-processor job while keeping the order of job starting times.

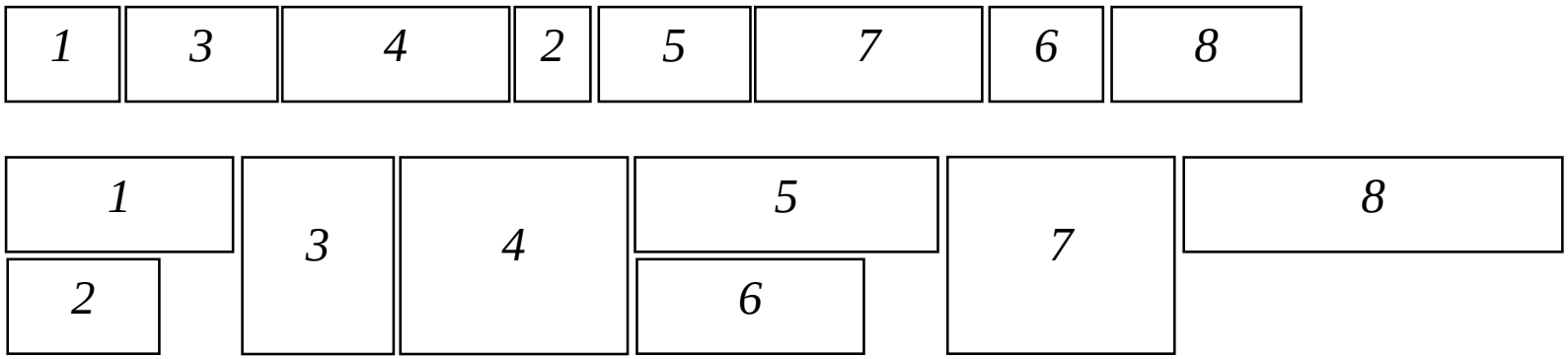
Rigid Jobs: Approximation Algorithm

Lower Bound

$\sum C_j^*(P1) \leq \sum C_j^*(P2)$

Theorem

A 2-approximate schedule can be found in $O(n\log n)$ time for scheduling problem $P2|size_j, energy|\sum C_j$.



Single Mode Jobs: $P2|fix_j, energy|\sum C_j$

Lower Bound

$\mathcal{J}' = \mathcal{J}_1 \cup \mathcal{J}_{12}$, $\mathcal{J}'' = \mathcal{J}_2 \cup \mathcal{J}_{12}$.

The first subproblem

$$\sum_{i=1}^{n'} (n' - i + 1) p'_{\pi_i} \to \min,$$

$$\sum_{i \in \mathcal{J}'} |fix_i| (p'_i)^{1-\alpha} W_i^{\alpha} \leq E.$$

$$C'_{\Sigma} = (E)^{1/1-\alpha} \left(\sum_{j=1}^{n'} W_{\pi_j} |fix_{\pi_j}|^{1/\alpha} (n' - j + 1)^{\alpha - 1/\alpha} \right)^{\alpha/\alpha - 1}.$$

The minimum sum of completion times is reached on the permutation, where the jobs are ordered by non-decreasing of $W_i |fix_i|^{1/\alpha}$.

Single Mode Jobs: Approximation Algorithm

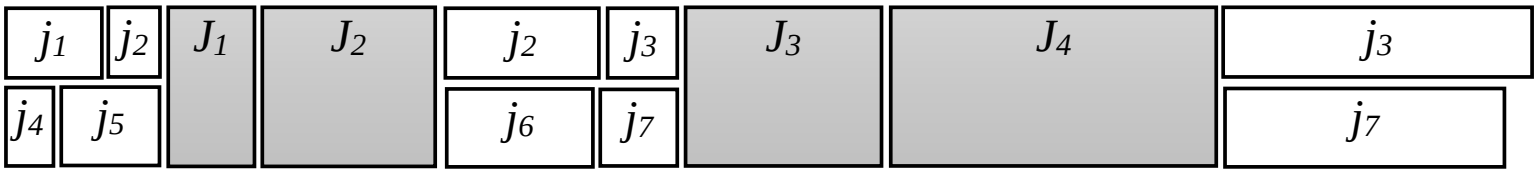
Step 1

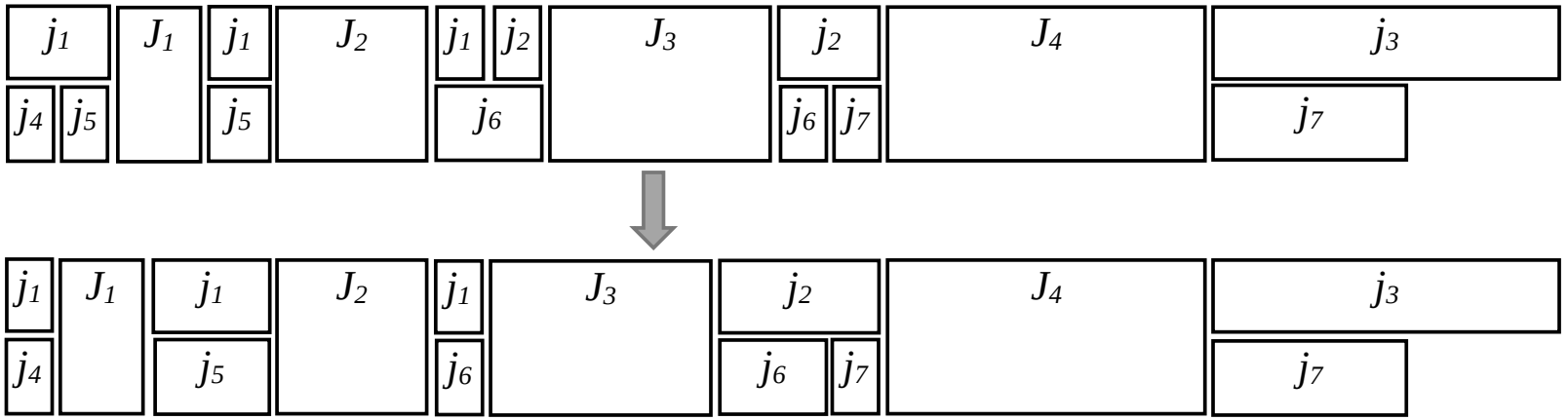
Decreasing the energy budget in both subproblems in two times, we obtain $2^{1/\alpha-1}$-approximate solutions S' and S''.

Step 2. Preemptive Schedule

Each two-processor job is executed without preemptions in interval $(\max\{C'_j, C''_j\} - \min\{p'_j, p''_j\},\ \max\{C'_j, C''_j\}]$.
Single-processor jobs are performed in the same order and with the same durations as in S' and S'', but may be preempted by two-processor jobs.
Reconstruct schedule such that at most one single-processor job is preempted by each two-processor job.
$C^{pr}_{\sum} \leq 2 \cdot 2^{1/\alpha-1} LB$.

Single Mode Jobs: Reconstruction





Single Mode Jobs: Approximation Algorithm

Step 3. Non-preemptive Schedule

Identify single-processor jobs $j_{i_1}, j_{i_2}, \ldots, j_{i_k}$ that are preempted by some two-processor jobs.
Change the start time of j_{i_l} to the completion time of the last two-processor job that preempts j_{i_l}.

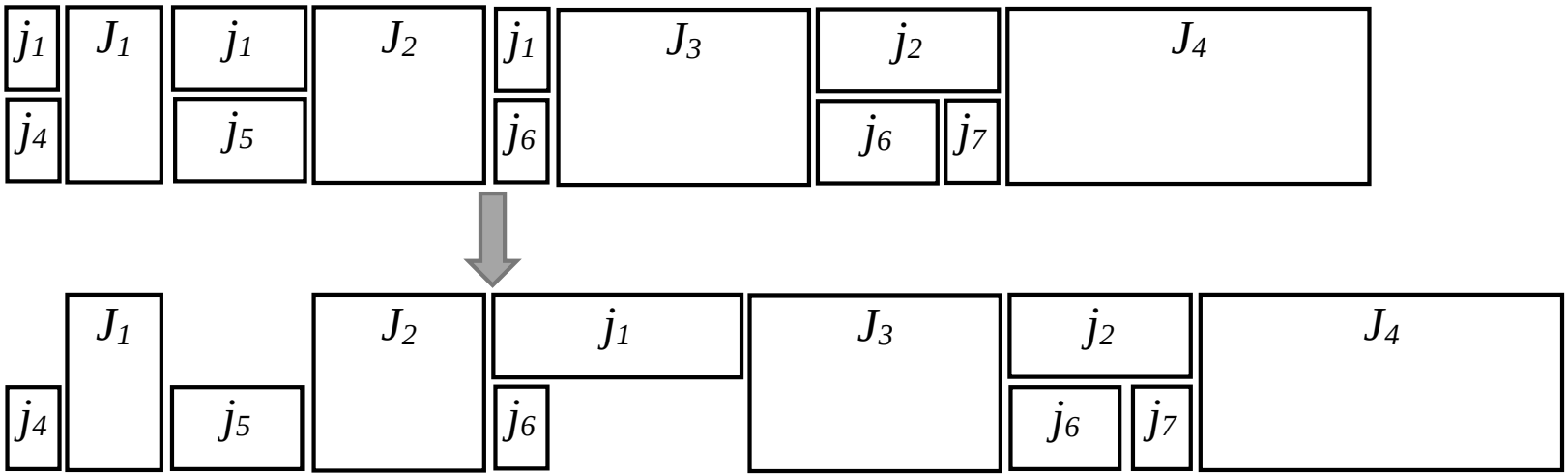
$$C_{\sum}^{npr} \leq 2C_{\sum}^{pr} \leq 2^{2\alpha-1/\alpha-1} LB.$$

Theorem

A $2^{2\alpha-1/\alpha-1}$-approximate schedule can be found in polynomial time for problem $P2|fix_j, energy|C_{\sum}$.

A $2^{\alpha/\alpha-1}$-approximate schedule can be found in polynomial time for problem $P2|fix_j, pmtn, energy|C_{\sum}$.

Single Mode Jobs: Non-preemptive Schedule



Obtained Results

Makespan

Problem	Complexity	Approx.
$P2\|size_j, energy\|C_{\max}$	NP-hard	$3/2$
$P2\|size_j, prec, energy\|C_{\max}$	NP-hard	2
$P2\|r_j, size_j, pmtn, energy\|C_{\max}$	$Poly$	–
$P2\|r_j, fix_j, pmtn, energy\|C_{\max}$	$Poly$	–
$P2\|fix_j, energy\|C_{\max}$	$Poly$	–

Total Completion Time

Problem	Complexity	Approx.
$P2\|size_j, energy\|\sum C_j$	?	2
$P2\|fix_j, energy\|\sum C_j$	?	$2\frac{2\alpha-1}{\alpha-1}$
$P2\|fix_j, pmtn, energy\|\sum C_j$	?	$2\frac{\alpha}{\alpha-1}$

Conclusion and Further Research

Conclusion

- NP-hardness and polynomial solvability of parallel and dedicated versions with makespan criterion and total completion time criterion.
- We propose approaches to construct approximation and exact algorithms for various particular cases.

Further Research

- The problems with more complex structure, where processors are heterogeneous and jobs have alternative execution modes with various characteristics.
- Open question is the complexity status of the problem with rigid and single mode jobs on two processors.

Thank you for your attention!