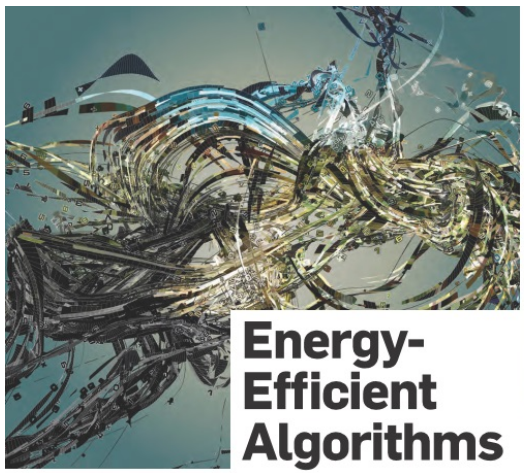
Адаптивные эвристики для одной задачи составления расписаний с учетом ресурсных ограничений

М.Ю. Сахно, Ю.В. Захарова

Омский филиал института математики
им. С.Л. Соболева СО РАН
Исследование выполнено за счет гранта Российского научного фонда № 22-71-10015, https://rscf.ru/project/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.



Report Structure

- Problem Statement
- Previous Research
- NP-hardness
- Greedy algorithms and lower bounds
- Genetic algorithm and experimental evaluation
- Further Research

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
$size_j$ is the number of processors required by job j
$W_j := \frac{V_j}{size_j}$ is the work on one processor

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 the minimum total completion time so that the energy consumption is not greater than a given energy budget.

Previous Research: Classic

Makespan

Drozdowski (2009): poly for parallel jobs, pmtn, r_j
approx for parallel jobs, r_j
Brucker (2000), Du, Leung (1989): parallel jobs: NP-hard, strongly NP-hard for prec

Total Completion Time

Lee and Cai (1999): parallel jobs: strongly NP-hard
Schwiegelshohn et. al. (1998), J. Turek et. al. (1994): approximation algorithms for parallel 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

Parallel jobs

Kononov, Zakharova (2017-2022): NP-hardness and approx
Kong F. et. al. (2011): level-packing algorithms
Li K. (2012): partitioning-scheduling-supplying

Convex Program (KKT-conditions)

Two-processor Jobs

$$\frac{1}{2}\sum_{i=1}^{n}(n-i+1)p_{\pi_i} \to \min,$$

$$\sum_{i=1}^{n}(V_{\pi_i})^{\alpha}p_{\pi_i}^{1-\alpha} = E.$$

Single-processor Jobs

$$\sum_{j\in\mathcal{J}} C_j(\pi) = \sum_{j=1}^{\frac{n}{2}}\left(\frac{n}{2}-j+1\right)(p_{\pi_{2j-1}}+p_{\pi_{2j}}) \to \min,$$

$$\sum_{j\in\mathcal{J}} {p_j}^{1-\alpha}(V_j)^{\alpha} = E.$$

NP-hardness

Even-Odd Partition Problem

$A = \{a_1, a_2, \ldots, a_{2n_0}\}$ is the ordered set such that
$\sum_{a_i \in A} a_i = 2C$, $a_i < a_{i+1}$, $i = 1, \ldots, 2n_0 - 1$
$a_{2i+1} > 3a_{2i}$ for $i = 1, \ldots, n_0 - 1$.
Question: whether A can be partitioned into two subsets A_1 and A_2

$$\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i = C, \ |A_1| = |A_2| = n_0,$$

A_1 contains only one element from each pair a_{2i-1}, a_{2i}, $i = 1, \ldots, n_0$.

Theorem

Problem $P2|size_j, energy| \sum C_j$ is NP-hard.

Algorithm 1

Scheme

Step 1: Given an instance I of $P2|size_j, energy|\sum C_j$, we generate the instance I' with fully-parallelizable jobs, construct optimal schedule S' for jobs, corresponding to non-decreasing order of volumes V_j, and find optimal durations p_j.
Step 2: Calculate processing times of jobs for instance I: $\frac{2p_j}{size_j}$, $j = 1, \ldots, n$. 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 jobs in non-decreasing of volumes V_j.

Lemma

$\sum C_j(S') \leq \sum C_j^*$.

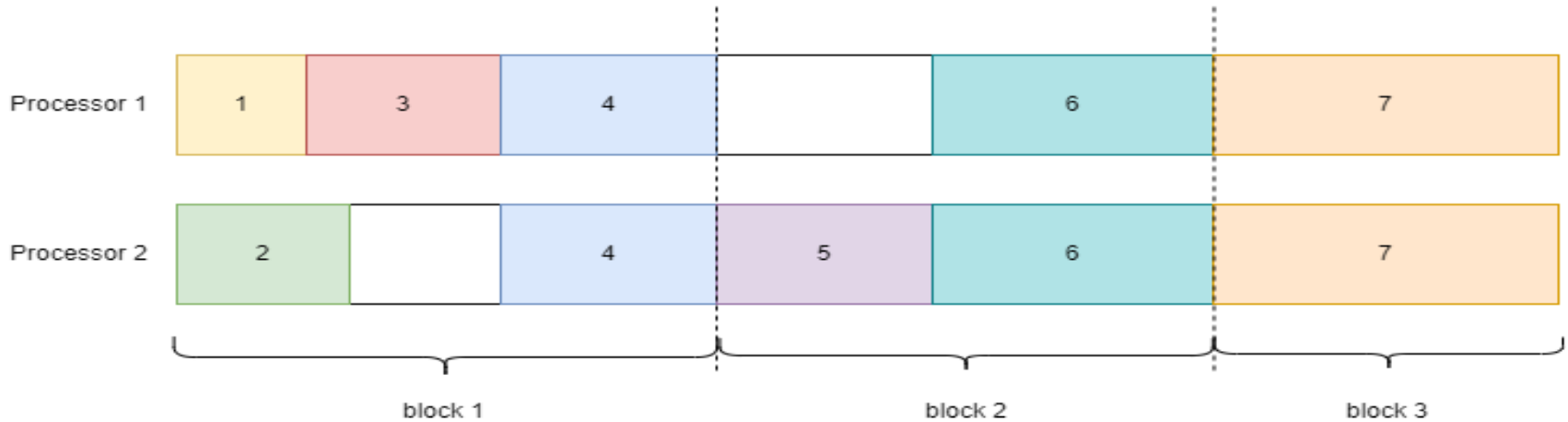
Theorem

A 2-approximate schedule can be found by Algorithm 1 in $O(nlogn)$ time for scheduling problem $P2|size_j, energy|\sum C_j$.

Local improvements

1. Find blocks

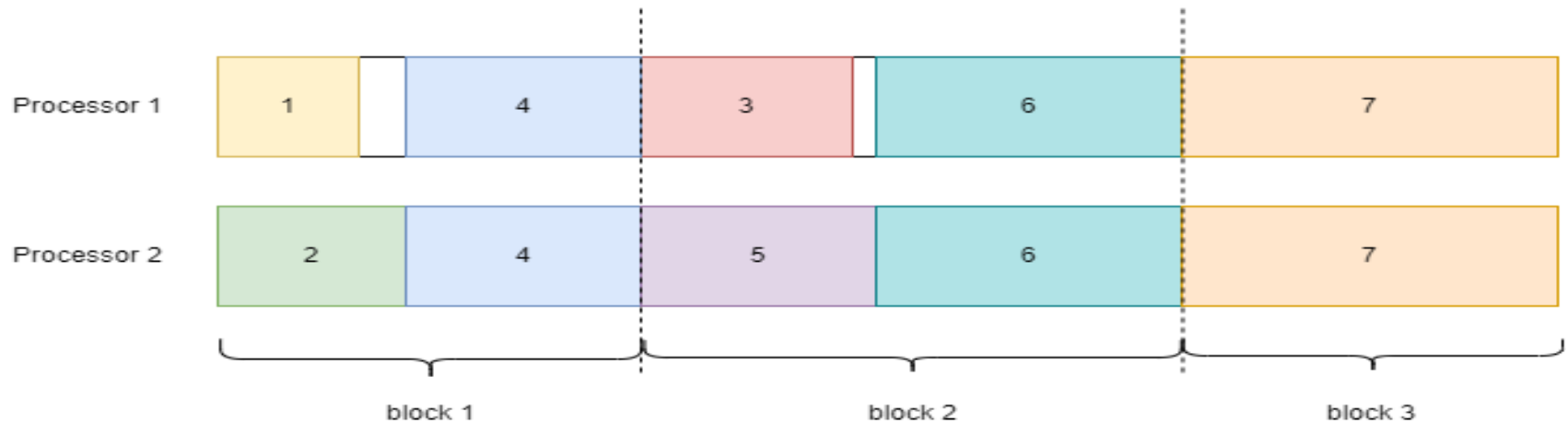
2. If a block consists of an odd number of single-processor jobs, move the last job to the next block if possible



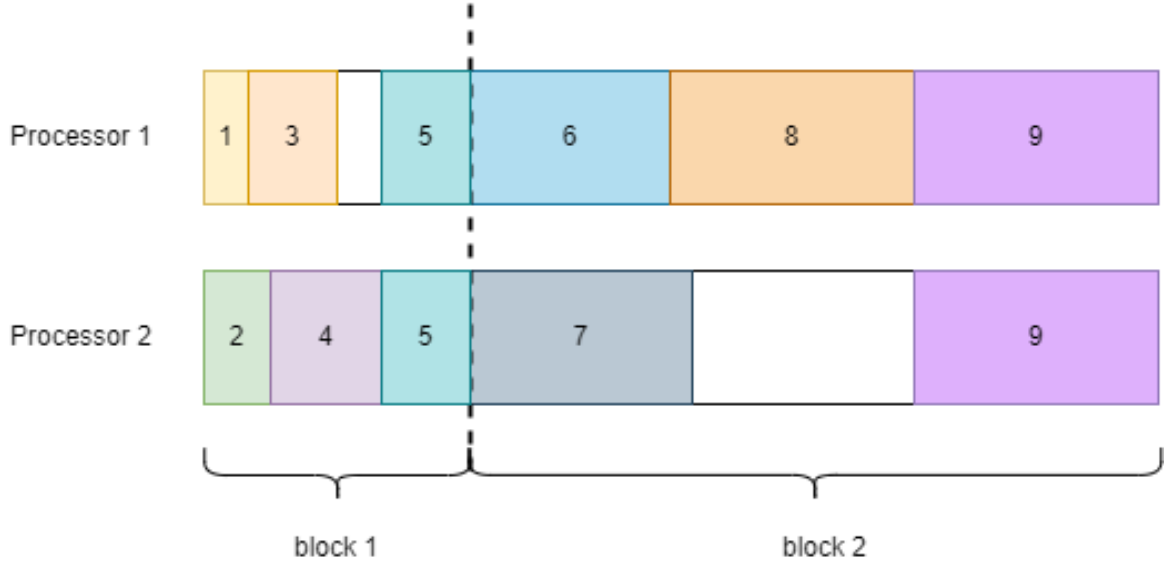
Local improvements

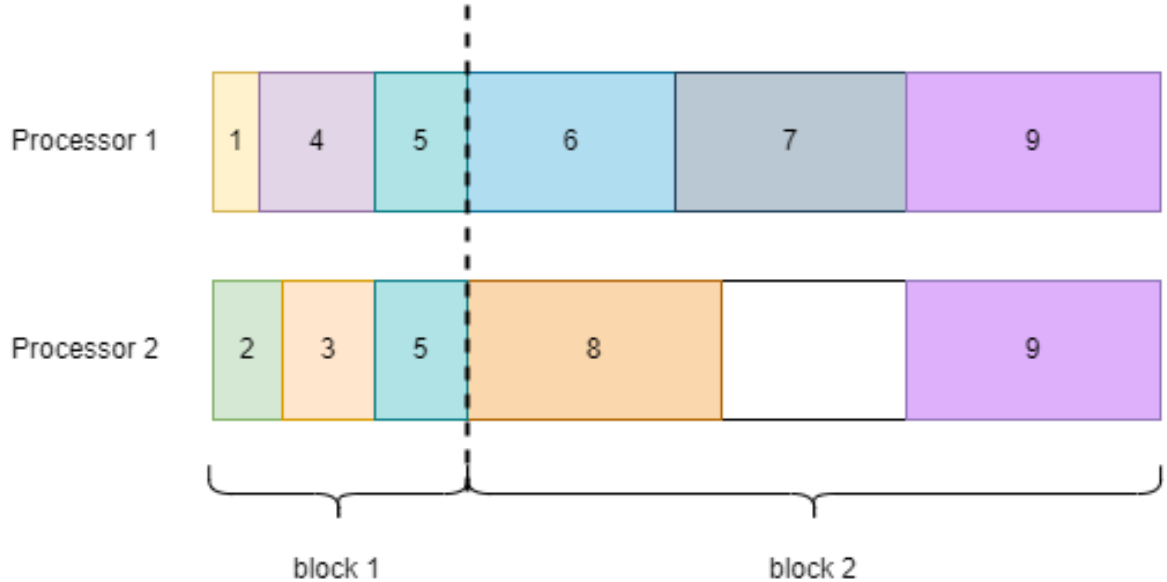
1. Find blocks

2. If a block consists of an odd number of single-processor jobs, move the last job to the next block if possible



Local improvements inside blocks





Algorithm 1 with Local Improvements (Algorithm 2)

Step 1. Construct a schedule by Algorithm 1 and find blocks in the solution.

Step 2. Consequently apply the local improvements between blocks.

Step 3. Apply local improvements inside blocks to the given solution.

Step 4. Return the found solution.

Genetic Algorithm

1: Construct the initial population of k permutations.
2: Until termination condition is met, perform
for $i \leftarrow 1$ to αk

2.1 Select two parent permutations π^1 and π^2.

2.2 Construct $(\pi^{1\prime}, \pi^{2\prime}) = Cross(\pi^1, \pi^2)$.

2.3 Apply insert mutation to permutations $\pi^{1\prime}$ and $\pi^{2\prime}$.

2.4 Compute the objective value of the offspring.

3: Return the best found solution.

Details

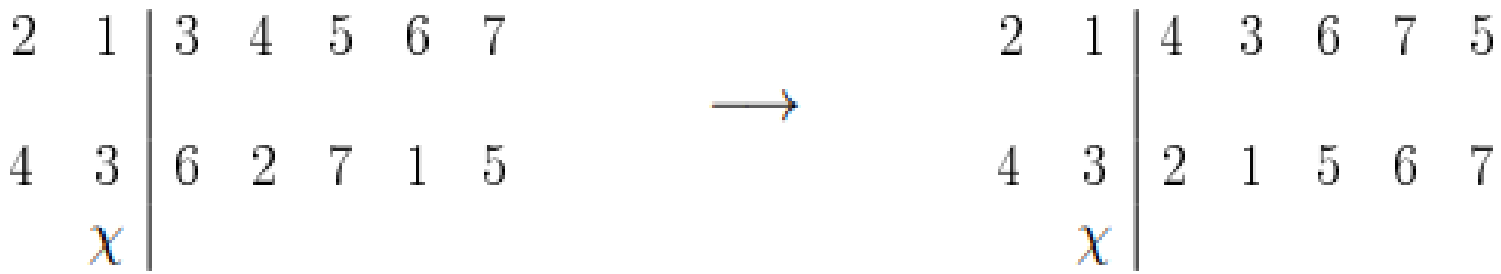
Initial population: problem specific constructive heuristic.

Selection: ranking.

Crossover: OPX.

Stopping criteria: quantity of fitness calculations.

We use the classic restarting rule.



OPX crossover

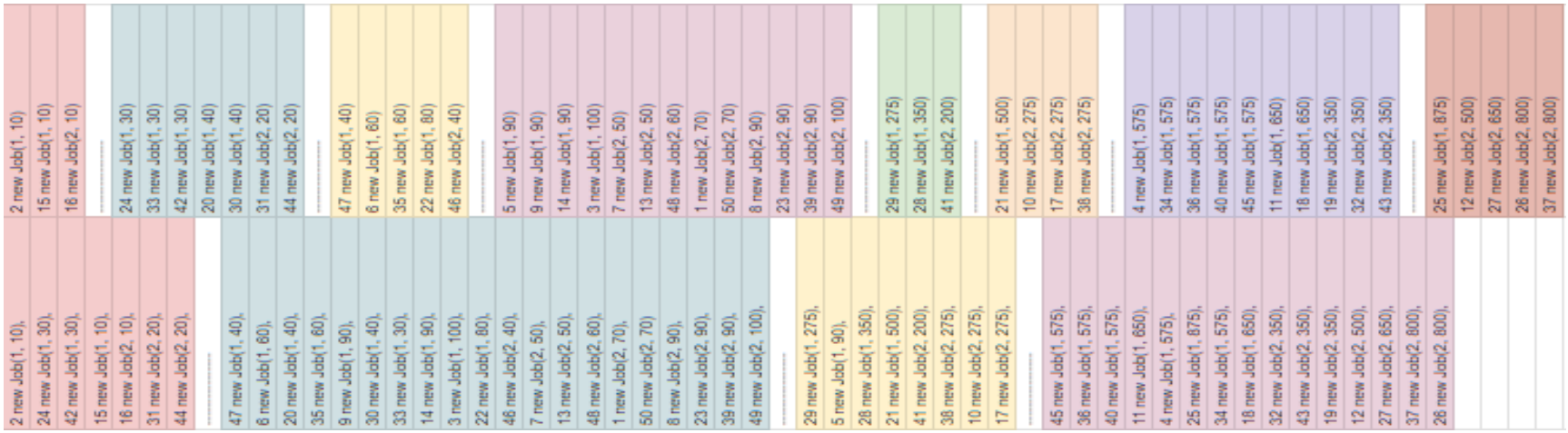
Experimental evaluation

num	A2	LB
1	0.63	1.04
2	0.80	1.97
3	0.85	3.11
4	0.94	1.39
5	1.05	1.31
6	1.15	0.94
7	1.27	1.61
8	1.27	2.63
9	1.37	1.18
10	1.59	1.21
11	1.70	2.14
12	1.75	1.90
13	1.89	2.68
14	1.99	1.56
15	2.43	2.89

num	A2	LB
16	2.47	2.22
17	2.68	1.60
18	2.73	2.90
19	3.00	1.82
20	3.27	2.90
21	3.39	2.07
22	3.72	0.84
23	3.84	2.45
24	3.90	1.89
25	3.92	2.26
26	3.93	3.67
27	3.96	2.16
28	4.13	2.25
29	4.32	1.63
30	4.39	2.73

Relative deviation in percentage of GA from Algorithm 2 solutions (A2) and lower bound (LB).

Experimental evaluation



Structure of solutions

Further Research

- More accurate selections in local improvements.
- Adaptive genetic algorithm.
- Optimized crossover operators.
- Adaptation of the GA to scheduling problems with different resources.
- Statistical analyzes of the experimental results.
- Theoretical analysis of the genetic algorithm.

Thank you for your attention!