

Evolutionary algorithm for speed scaling scheduling problem

Yulia Zakharova

The research was supported by Russian Science Foundation grant
N 22-71-10015

OPTIMA-2023

Sobolev Institute of Mathematics, Omsk, Russia

Report structure

- ▶ Problem Statement
- ▶ Previous Research and Our Results
- ▶ Local Search
- ▶ Evolutionary Algorithms
- ▶ Computational Experiment

$1|r_j, d_j|E$

Input Data

$J = \{1, \dots, n\}$ is the set of jobs.

W_j is the volume of job j .

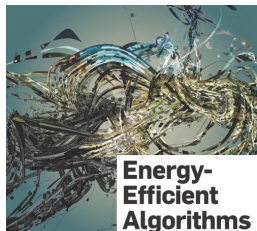
r_j is the release date of job j .

d_j is the deadline of job j .

Preemptions are disallowed.

Agreeable Release Dates and Deadlines

For any two jobs i and j , relation $r_i < r_j$ implies $d_i \leq d_j$.

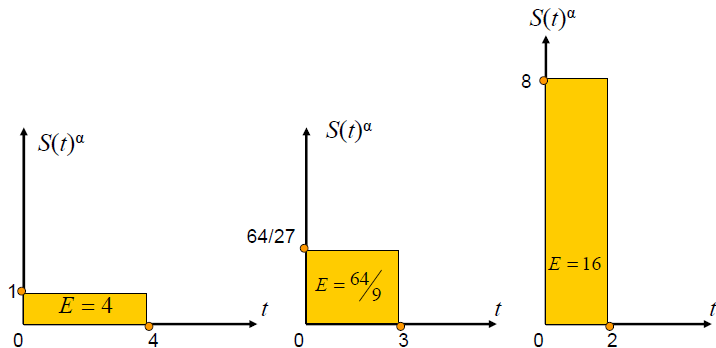


Homogeneous Model in Speed-scaling

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.

The objective is to find a feasible schedule that minimizes the total energy consumption.



Related Results: Algorithms

Energy-Efficient Scheduling for Parallel Real-Time Tasks Based on Level-Packing

Kong F. et. al. (SAC'11): two-dimensional strip packing problem, energy consumption assignment

Energy efficient scheduling of parallel tasks on multiprocessor computers

Li K. (Journal of Supercomputing, 2012): system partitioning, task scheduling, power supplying

A genetic algorithm for energy-efficiency in job-shop scheduling

Salido M.A. (Int. J. Adv. Manuf. Technol., 2015): genetic algorithm with generational scheme – position-based encoding, problem specific initial population, order crossover, shuffle mutation.

Related Results: Complexity

$1|pmtn, r_j, d_j|E$ and $P|agree, r_j, d_j|E$

Yao, Demers, Shenker (1995): $O(n^2)$ time;

Shioura, Shakhlevich, Strusevich (2015): $O(n^3)$ time.

$1|r_j, d_j|E$

Antoniadis, Huang (2013): NP-hard, $2^{5\alpha-4}$;

Bampis, Kononov, Letsios et. al. (2018): $2^{\alpha-1}(1+\varepsilon)^\alpha \tilde{B}^\alpha$.

Preemptive and Agreeable instances

Algorithm 1 YDS Algorithm (Yao, Demers, Shenker), 1995

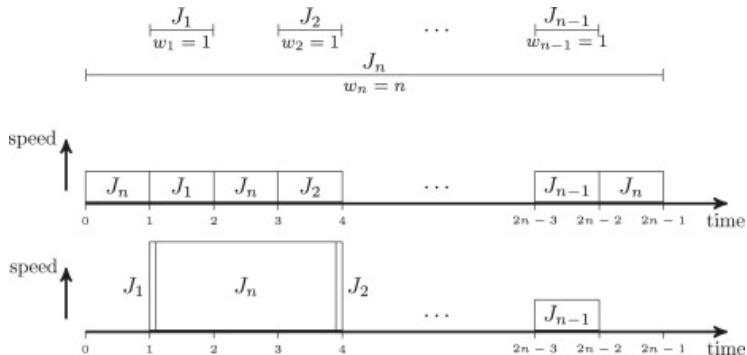
1: While $\mathcal{J} \neq \emptyset$:

1.1 Let $[t, t')$ be the interval with maximum density, i.e., that maximizes $\frac{\sum_{j \in J(t, t')} W_j}{t' - t}$.

1.2 Process jobs $i \in J(t, t')$ in interval $[t, t')$ using the earliest deadline policy with speed equal to the maximum density. Then remove the jobs $J(t, t')$ from \mathcal{J} , and adjust the remaining jobs as if the time interval $[t, t')$ does not exist.

2: Return the resulting schedule and its objective value.

Preemptive vs Non-Preemptive



Small Neighborhoods

Solution encoding

Solutions are encoded as permutations.

Consider a pair of indexes $i < j$ and correct release dates and deadlines as follows: $r'_j = \max\{r_i, r_j\}$ and $d'_i = \min\{d_i, d_j\}$.

Objective value may be calculated in $O(n^2)$ time for the given permutation.

Neighborhoods

Swap neighborhood: positions of two jobs are exchanged.

Insert neighborhood: inserting a job in some other position.

Partial Order Between Jobs

Release dates and deadlines give us a partial order between jobs: if $d_i < r_j$ then job i must precedes job j .

We exchange only independent jobs in the neighborhoods.

Large Neighborhoods

Optimal Recombination Problem (ORP)

Given two parent solutions π^1 and π^2 . It is required to find a permutation π' such that:


- (I) $\pi'_i = \pi_i^1$ or $\pi'_i = \pi_i^2$ for all $i = 1, \dots, n$;
- (II) π' has the minimum value of objective function $E(\pi')$ among all permutations that satisfy condition (I).

Optimal recombination may be considered as a best-improving move in a large neighbourhood defined by two parent solutions.

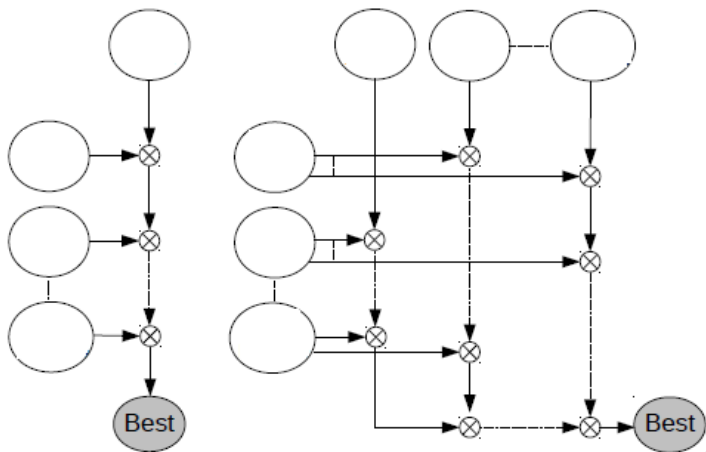
The ORP is NP-hard, but “almost all” instances are polynomially solvable.

Population Local Search (PLS)¹ [NUMTA2023]

- 1: Construct the initial population of m permutations (feasible in accordance with release dates and deadlines).
 - 2: Apply local search based on swap or insert neighborhood to each permutation.
 - 3: For $j=1$ to m perform Steps 4-6:
 - 4: Generate random sequence of permutations π^1, \dots, π^m .
Put $\pi' = OR(\pi^1, \pi^2)$.
 - 5: For $i=3$ to m construct
$$\pi' = OR(\pi', \pi^i)$$
 - 6: Improve π' by local search withing swap or insert neighborhood and save as π'_j .
 - 7: Put $\pi'' = OR(\pi'_1, \pi'_2)$.
 - 8: For $j=3$ to m construct
$$\pi'' = OR(\pi'', \pi'_j)$$
 - 9: Return π'' and $E(\pi'')$.
-

¹R. Tinos, D. Whitley, G. Ochoa (2020): A New Generalized Partition Crossover for the Traveling Salesman Problem: Tunneling Between Local Optima 

Population Local Search (PLS) [NUMTA2023]

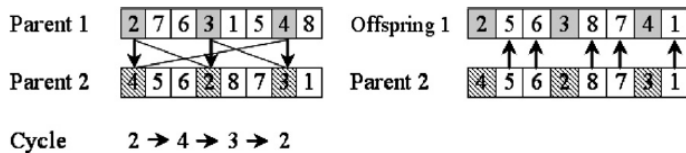


Genetic Algorithm with OR ($GA : ORP$) [NUMTA2023]

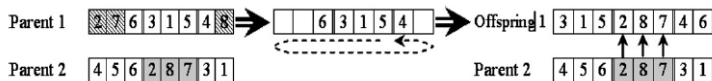
-
- 1: Construct the initial population of m permutations (feasible in accordance with release dates and deadlines).
 - 2: Apply local search based on swap or insert neighborhood to each permutation.
 - 3: Until termination condition is met, perform
 - 3.1 Select two parent permutations π^1 and π^2 .
 - 3.2 Apply swap or insert mutation to permutations π^1 and π^2 .
 - 3.3 Put $\pi' = OR(\pi^1, \pi^2)$.
 - 3.4 Replace the worst permutation of the population by π' .
 - 4: Improve the record solution by local search withing swap or insert neighborhood.
 - 5: Return the best found solution.
-

Crossover Operators

Cycle Crossover (CX)

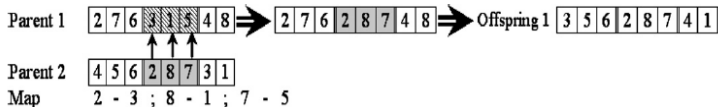


Order Crossover (OX)

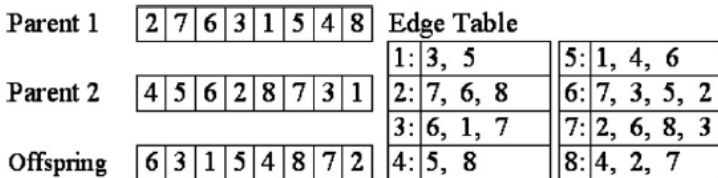


Crossover Operators

Partially Mapped Crossover (PMX)



Edge Recombination (EX)



Genetic Algorithm with Generational Scheme (GAGs)

- 1: Construct the initial population of m permutations.
- 2: Apply local search based on swap or insert neighborhood to each permutation.
- 3: Until termination condition is met, perform
 - for $i \leftarrow 1$ to βm
 - 2.1 Select two parent permutations π^1 and π^2 .
 - 2.2 Construct $(\pi^{1'}, \pi^{2'}) = \text{Cross}(\pi^1, \pi^2)$.
 - 2.3 Apply insert mutation to permutations $\pi^{1'}$ and $\pi^{2'}$.
 - 2.4 Compute the objective value of the offspring.
- 4: Return the best found solution.

Adaptive Technique

$$\phi_a = \begin{cases} w_1, & \text{if the new solution is a new global best,} \\ w_2, & \text{if the new solution is better than the current one,} \\ w_3, & \text{if the new solution is accepted,} \\ w_4, & \text{if the new solution is rejected.} \end{cases}$$

$$\rho_a = \lambda\rho_a + (1 - \lambda)\phi_a.$$

Computational Experiment: Input Data

Number of jobs $n = 50$ and $n = 100$.

Parameter $\alpha = 2$ and $\alpha = 3$.

Release date r_j is selected randomly from interval $[0, 20]$.

Deadline d_j is generated randomly from $[r_j + 1, r_j + 11]$.

Volume W_j is chosen randomly from $[5, 15]$.

Computational Experiment: Relative Deviation from Lower Bound

Series	<i>PLS</i>			<i>GA : ORP</i>		
	min	aver	max	min	aver	max
$S_{\alpha=2, n=50}$	0	0,9	6,1	0	1,7	8,2
$S_{\alpha=3, n=50}$	0	2,1	10,1	0	3,2	12,7
$S_{\alpha=2, n=100}$	1,7	3,1	6,1	2,8	5,2	7,9
$S_{\alpha=3, n=100}$	0,5	5,8	15,1	0,8	8,8	17,3

Series	<i>GA : Adapt</i>			<i>GA : PMX</i>		
	min	aver	max	min	aver	max
$S_{\alpha=2, n=50}$	0	0,6	3,1	0	0,7	3,7
$S_{\alpha=3, n=50}$	0	1,9	9,5	0	2,0	9,7
$S_{\alpha=2, n=100}$	1,5	3,1	5,5	1,7	3,1	5,9
$S_{\alpha=3, n=100}$	0,3	5,7	12,5	0,3	5,9	12,3

Series	<i>GA : Mut</i>		
	min	aver	max
$S_{\alpha=2, n = 50}$	0	1,9	9,0
$S_{\alpha=3, n = 50}$	0	3,9	9,8
$S_{\alpha=2, n = 100}$	3,2	6,0	9,5
$S_{\alpha=3, n = 100}$	2,1	9,1	12,7

Speed Scaling Scheduling: $P2|size_j, energy| \sum C_j$

Processors and Jobs

$m = 2$ speed-scalable processors

$\mathcal{J} = \{1, \dots, n\}$ 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:

$$P2 |size_j, energy| \sum C_j$$

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

Experimental Results: $P2|size_j, energy| \sum C_j$

num	GA	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	GA	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 with 1-OX from Greedy Algorithm solutions (GA) and lower bound (LB).

Conclusion and Further Research

- ▶ We proposed and investigated the evolutionary algorithm with various operators and schemes for the single processor speed scaling scheduling problem.
- ▶ Experimental evaluation on instances of different structures shown that the algorithms demonstrate competitive results.
- ▶ Further research can be undertaken to various optimized and/or randomized recombination operators and comparison of them in context of the presented algorithms and their composition; generalize to the case of several processors.

Thank you for your attention!