Investigation of operators and parameters in evolutionary algorithms for one scheduling problem with resource constraints

Maria Sakhno

Omsk Department of Sobolev Institute of Mathematics SB RAS
The research is supported by RSF grant 22-71-10015.

Speed Scaling Scheduling

Processors and Jobs

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
E is the energy budget

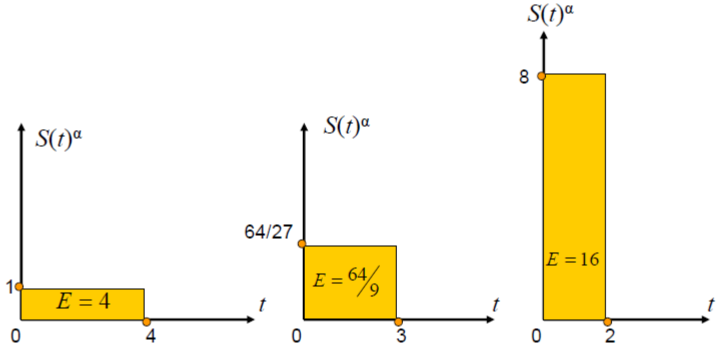
Parameters

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

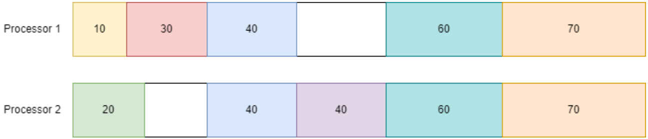
If a processor runs at speed s then the energy consumption is s^α units of energy per time unit, where $\alpha > 1$ is a constant (practical studies show that $\alpha \leq 3$).

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



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.

Solution



Lower Bound

Previous Research

Scheduling

- Lee & Cai: Scheduling one and two-processor tasks on two parallel processors (1999)
- Kononov & Zakharova: Speed scaling scheduling of multiprocessor jobs with energy constraint and total completion time criterion (2023)
- Zakharova & Sakhno: Heuristics with local improvements for two-processor scheduling problem with energy constraint and parallelization (2024)

Evolutionary Computation

- Eremeev & Kovalenko: A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem (2020)
- Neri & Cotta: Memetic Algorithms and Memetic Computing Optimization: A Literature Review (2012)
- Blum & Eremeev & Zakharova: Hybridizations of evolutionary algorithms with Large Neighborhood Search (2022)
- Doerr & Ghannane, & Ibn Brahim: Runtime Analysis for Permutation-based Evolutionary Algorithms (2024)

Genetic Algorithm (GA) with Generational Scheme

1: Construct the initial population $P^0 = \{\pi_j^0\}$ of k permutations. Save n_e individuals with the best objective values as elites of P^0. Put $t = 0$.

2: Until termination condition is met, perform

2.1 for $i \leftarrow 1$ to $(k - n_e)/2$

2.1.1 Select two parent permutations π^1 and π^2 using operator $Sel(P^t)$.

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

2.1.3 Apply the mutation operator to constructed permutations: $Mut(\pi^{1\prime})$ and $Mut(\pi^{2\prime})$ and save the result as individuals $\pi_{2i-1}^{t+1}, \pi_{2i}^{t+1}$ for population P^{t+1}.

2.2 Copy elites of P^t to P^{t+1} and identify elites of P^{t+1}.

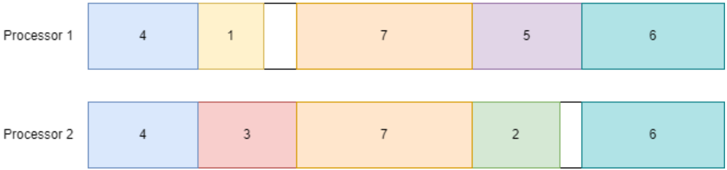
2.3 Put $t = t + 1$.

3: Return the best found individual.

Solution encoding

The solutions are encoded by permutations of jobs.

4, 1, 3, 7, 5, 2, 6



Crossover Operators

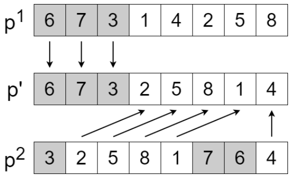


Figure: One Point Crossover (1PX)

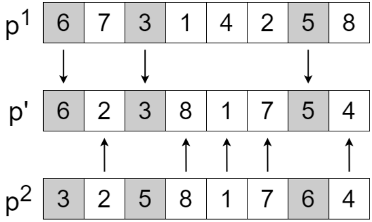


Figure: Cycle Crossover (CX)

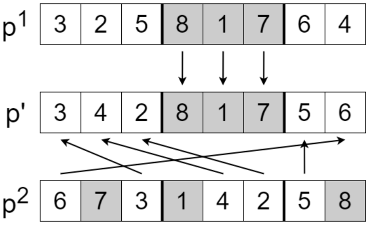


Figure: Order Crossover (OX)

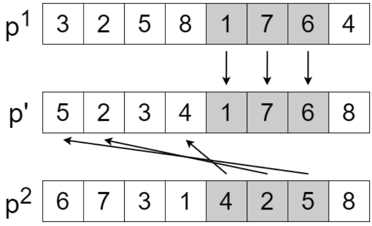
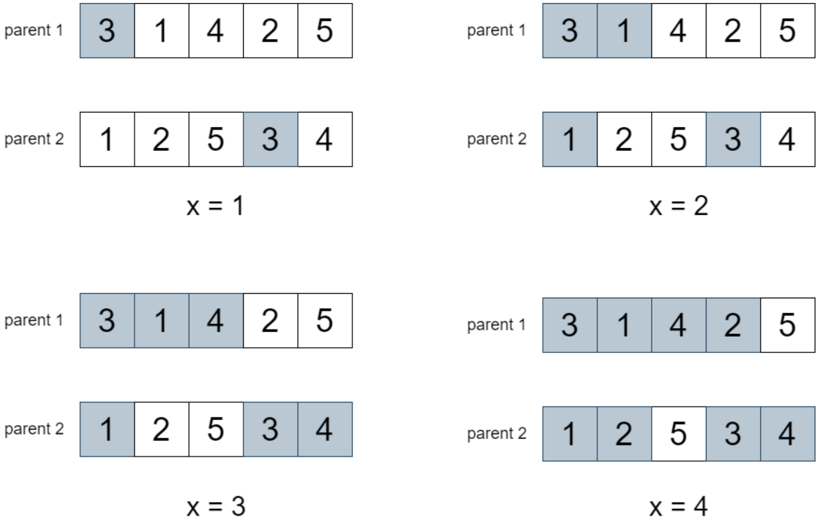


Figure: Partially Mapped Crossover (PMX)

Optimized Crossovers

One Point Crossover (1PX)



Mutation Operators

Exchange (swap) mutation

3	8	4	1	5	2	7	6	⟹	3	8	7	1	5	2	4	6

Shift (insert) mutation

3	8	4	1	5	2	7	6	⟹	3	8	4	5	2	7	1	6

Scramble Mutation Scheme[1]

1. Randomly choose n_p from Poisson distribution with λ_p.
2. Apply operator Mut for the given genotype n_p times.

[1] Doerr & Ghannane, & Ibn Brahim: Runtime Analysis for Permutation-based Evolutionary Algorithms (2024)

Adaptive Technique[2]

1: Choose a crossover. The probability of choosing each operator is proportional to its weight.
2: Apply chosen crossover to the parent genotypes.
3: Update the weight of the chosen crossover:

$$\phi_a = \begin{cases} w_1, \text{if the new solution is a new global best,} \\ w_2, \text{if the new solution is better than the current one,} \\ w_3, \text{if the new solution is better than one of the parents or both.} \end{cases}$$

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

[2] Mara & Norcahyo & Jodiawan & Lusiantoro & Rifai: A survey of adaptive large neighborhood search algorithms and applications (2022)

Parameter auto-tuning: IRACE package[3]

Parameter name	Parameter description
k	population size
n_e	number of elites
P_{IPRand}	probability of generating a genotype randomlly
$Selection$	selection operator
P_{Cross}	probability of applying the crossover operator
$Crossover$	crossover operator
P_{Mut}	probability of applying the mutation operator
$Mutation$	mutation operator
w_2, w_3, λ	parameters of adaptive technique
λ_p	lambda for Poisson distribution

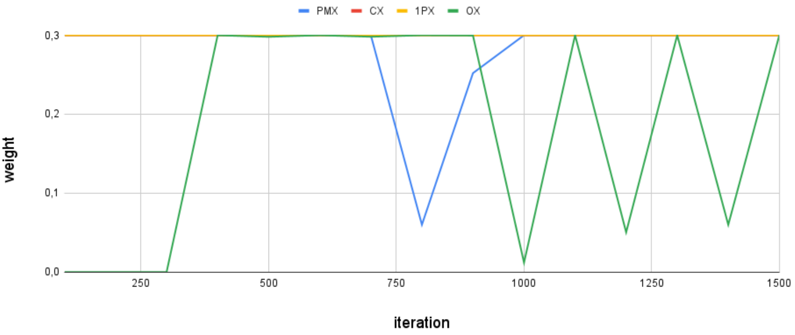
[3] Lopez-Ibanez, M., Dubois-Lacoste, J., Perez Caceres, L, Birattari, M., Stutzle, T.: The irace package: Iterated racing for automatic algorithm configuration, Operations Research Perspectives, 3, 43-58 (2016)

Versions of genetic algorithm

GA_{rand} is a GA.

GA_{adapt_rand} is the GA with the adaptive technique for randomized crossover operators ($1PX$, CX, OX, PMX).

Dynamics of crossover weights during GA_{adapt_rand} iterations



The classic restarting rule is used.

Versions of genetic algorithm

GA_{rand} is a classic GA.

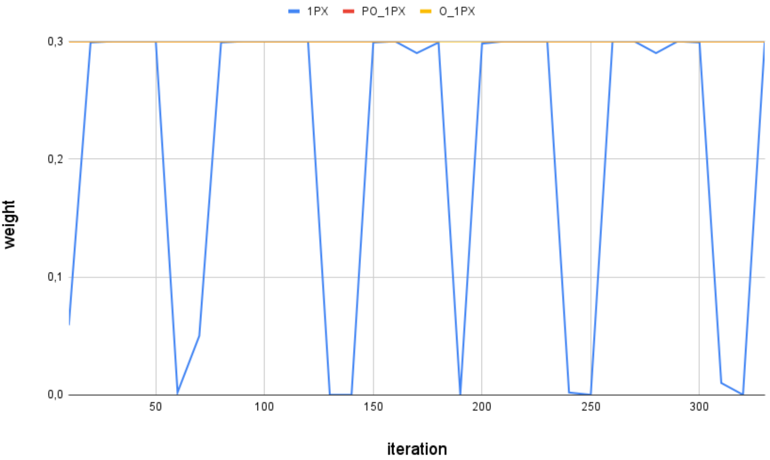
GA_{adapt_rand} is the GA with the adaptive technique for randomized crossover operators ($1PX$, CX, OX, PMX).

GA_{adapt_opt} is the GA with the adaptive technique for optimized crossover operators (1PX, PO_1PX, O_1PX).

GR_{LI} is the known greedy heuristic with local improvements[4].

[4] Zakharova & Sakhno: Heuristics with local improvements for two-processor scheduling problem with energy constraint and parallelization (2024)

Dynamics of crossover weights during GA_{adapt_opt} iterations



The classic restarting rule is used.

Experiment result

	GA_{rand}	GA_{adapt_rand}	GA_{adapt_opt}	GR_{LI}
avg	1.99%	2.05%	1.94%	4.56%
min	0.82%	0.83%	0.81%	1.67%
max	3.86%	3.76%	3.63%	7.74%

Table: Relative deviations of results from the lower bound for algorithms with parameters found by IRACE package

	$GA_{rand\_poisson}$	$GA_{adapt\_rand\_poisson}$	$GA_{adapt\_opt\_poisson}$
avg	1.96%	1.96%	1.95%
min	0.83%	0.86%	0.8%
max	3.72%	3.57%	3.63%

Table: Relative deviations of results from the lower bound for algorithms with scramble mutation operator and with parameters found by IRACE package

Conclusions and Further Research

Recommendations

- Apply auto-tuning for parameters of algorithm.
- Apply adaptive technique to identify the leading crossover operator.
- Implement optimized version of the leading crossover operator and try to apply scramble mutation.

Further Plans

- Generalize the algorithm on permutation problems.
- Compare with other known algorithms
  (P.A. Borisovsky, "A parallel "Go with the winners" algorithm for some scheduling problems", 2023;
  P. Borisovsky, Y. Kovalenko, "A Memetic Algorithm with Parallel Local Search for Flowshop Scheduling Problems", 2020).

Thank you for your attention!

Convex program

$$\sum_{j\in\mathcal{J}} C_j(\pi) = \sum_{j=1}^{n}(n-j+1)p_{\pi_j} \to \min, \qquad (1)$$

$$\sum_{j\in\mathcal{J}} (2p_j)^{1-\alpha}(V_j)^{\alpha} = E. \qquad (2)$$

Experiment result

	GA_{rand}	GA_{adapt_rand}	GA_{adapt_opt}	GR_{LI}
avg	2.03%	2.06%	1.95%	4.56%
min	0.83%	0.83%	0.78%	1.67%
max	3.83%	3.88%	3.57%	7.74%

Table: Relative deviations of results from the lower bound for algorithms

	GA_{rand}	GA_{adapt_rand}	GA_{adapt_opt}	GR_{LI}
avg	1.99%	2.05%	1.94%	4.56%
min	0.82%	0.83%	0.81%	1.67%
max	3.86%	3.76%	3.63%	7.74%

Table: Relative deviations of results from the lower bound for algorithms with parameters found by IRACE package

	$GA_{rand\_poisson}$	$GA_{adapt\_rand\_poisson}$	$GA_{adapt\_opt\_poisson}$
avg	1.96%	1.96%	1.95%
min	0.83%	0.86%	0.8%
max	3.72%	3.57%	3.63%

Table: Relative deviations of results from the lower bound for algorithms with scramble mutation operator and with parameters found by IRACE package