S©MR

СИБИРСКИЕ ЭЛЕКТРОННЫЕ
МАТЕМАТИЧЕСКИЕ ИЗВЕСТИЯ
Siberian Electronic Mathematical Reports
http://semr.math.nsc.ru
ISSN 1813-3304

Vol. 22, No. 2, pp. C85–C103 (2025) УДК 519.8
https://doi.org/10.33048/semi.2025.22.C07 MSC 90-08

EVOLUTIONARY ALGORITHM WITH OPTIMIZED OPERATORS FOR MULTI-PROCESSOR SCHEDULING WITH RESOURCES

Y.V. ZAKHAROVA, M.Y. SAKHNO

Communicated by the Programme Committee of the XXIV International Conference "Mathematical Optimization Theory and Operations Research" (MOTOR 2025)

Abstract: We consider the NP-hard multiprocessor scheduling problem with renewable and non-renewable resources that influence on the processing times of jobs. We propose an evolutionary algorithm with optimized operators and problem specific encoding scheme. The optimal recombination problem is solved in the crossover operator. The computational complexity of this problem and the convergence of the algorithm are analyzed. An experimental evaluation on real data shows that the algorithm demonstrates competitive results. A mixed integer programming model is also proposed and its properties are investigated.

Keywords: schedule, metaheuristic, computational complexity, experiment.

Zakharova, Yu.V., Sakhno, M.Y., Evolutionary Algorithm with Optimized Operators for Multi-Processor Scheduling with Resources.

© 2025 Zakharova Yu.V., Sakhno M.Y.

The work is supported by the Mathematical Center in Akademgorodok under the agreement № 075-15-2025-349 with the Ministry of Science and Higher Education of the Russian Federation.

Received June, 30, 2025, Published November, 24, 2025.

1 Introduction

Modern manufacturing and computing systems provide new scheduling problems where machines correspond to processors or processor cores and the consumption of renewable and non-renewable resources is taken into account. Renewable resources are given by limits at each time moment with subsequent renewal at the next time moment (e.g., cache memory, data bus). Non-renewable resources are limited for a certain period and distributed among jobs at different time moments (e.g., energy, finance for rent). Note that the duration of jobs may depend on the volumes of resource consumption.

We consider the multi-machine scheduling problem with resource-dependent processing times of jobs. A set of jobs $J = \{1, ..., n\}$ must be scheduled on a set of parallel machines (processors) $I = \{1, ..., m\}$. Each job $j \in J$ is characterized by the processing volume W_j and the number of the required processors (size) $size_j \le m$. Note that W_j is the execution time of job $j \in J$ on one processor with unit speed and without other jobs. Preemptions are disallowed. A partial order is given on the set of jobs. If job i precedes job j (denoted by $i \to j$), then job j can be started only when job i is completed.

The real durations of jobs depends on resource consumption. We consider two types of resources: renewable and non-renewable.

Energy represents the non-renewable resource, that depends on processor speed, and the speed influences on processing time of jobs. If job j is performed at speed s_j, then the duration $p_j = \frac{W_j}{s_j}$ and energy consumption $E_j = size_j \frac{W_j}{s_j} s_j^{\alpha}$, where $\alpha > 1$ is a constant. The total energy budget is given by the value E.

A data bus is a part of the system bus that is used to transfer data between computer components. The data bus capacity corresponds to renewable resources. The parallel execution of jobs and their joint consumption of the data bus influences the speed and duration of the jobs. In practice, the distribution of data bus among the threads is very hardware-specific and depends on many factors, which we can not afford to take into account (see e.g. [18]). As a simple approximation, we assume that the data bus bandwidth allocation to jobs may be found by the following rule. Let b_j denote the percent of data bus consumed by job $j \in J$. Suppose that l jobs are executed simultaneously, denote this set of jobs by J_s. Initially we assign to each job $b'_j := \frac{100\%}{l}$ of the data bus. If a job j has $b_j < b'_j$ then the remaining resource $(b'_j - b_j)$ is divided between jobs $i \in J_s$, for which $b_j > b'_j$. The processing time of job $j \in J_s$ is equal to $\frac{b'_j}{b_j} p_j$. We assume that slowing induced by the data bus does not influence energy consumption.

It is required to construct a feasible schedule w.r.t. allocation jobs to processors, partial order, energy constraint and common utilization of the data bus bandwidth, that minimizes one of the objectives. Let C_j denote the completion time of job $j \in J$ in a feasible schedule. We consider two criteria: the makespan $\max_{j \in J} C_j$ (the completion time of the last job) and the total completion time $\sum_{j \in J} C_j$.

Using the reduction from the Partition, 3-Partition, 3-Satisfiability and Linear Arrangement problems [13, 14, 5] we can prove NP-hardness for both considered criteria (the makespan and the total completion time) in the cases:
– arbitrary resource consumption, unit volumes, two processors, empty partial order (ordinary NP-hard);
– arbitrary resource consumption, unit volumes, three processors, empty partial order (strongly NP-hard);
– identical resource consumption, arbitrary volumes, at most two processors, non-empty partial order (strongly NP-hard).

Sequential and parallelizable jobs were investigated for settings with energy resource. NP-hard and polynomially solvable cases have been identified and polynomial time algorithms with approximation guarantees have been provided (see, e.g., [11, 12, 6]). In [21], a mixed integer programming (MIP) model is provided. Heuristics based on greedy, packing, local search and evolutionary schemes were proposed and experimentally tested in [10, 15, 20, 23].

A configuration MIP model and a greedy algorithm were proposed for scheduling processing modules with respect to data bus bandwidth and tested on small-size instances (up to 10 modules) [4]. In [8], it is assumed that the amount of resources allocated to each job is limited and continuous, and it is determined by the scheduler at each time moment. The heuristics proposed in [16, 24] are based on the principle that jobs should be allocated to processor cores in a complementary manner so that jobs with the most diverse resource usage needs are executed simultaneously. Such resources are data bus bandwidth and cache at different levels.

Our main result is an evolutionary algorithm with optimized operators and encoding-decoding scheme allowing to take into account resources: energy and data bus bandwidth. The complexity status and algorithmic aspects of operators are investigated and the convergence of the evolutionary algorithm is analyzed. A mixed integer programming model of the problem is also presented.

2 Mixed Integer Programming Model

Here we provide the mathematical model of the problem with discrete set of possible speeds of jobs based on the concept of configurations and event points.

A configuration c is the feasible subset of jobs that can be executed simultaneously, joined with their speeds. Let C denote the set of all configurations. Configurations are formed in accordance with the partial order, and we have a partial order on the set of configurations ($a_{c_1,c_2} > 0$ indicates that c_1 precedes c_2).

By *event point* we will mean a subset of variables in mixed integer linear programming model, which characterize a selection of a certain set of jobs and their starting and completion times, as well as resource consumption. Let $K = \{1, \dots, k_{\max}\}$ denote the set of event points.

The following parameters are determined and calculated at the preprocessing stage: s_{jc} is the speed of job j in configuration c, $E_c = \sum_{j \in C} s_{jc}^{\alpha}$ is the instantaneous power for configuration c and

$$q_{jc} = \begin{cases} 1, & \text{if job } j \text{ is performed in configuration } c, \\ 0 & \text{otherwise.} \end{cases}$$

Introduce variables:

$$x_{kc} = \begin{cases} 1, & \text{if configuration } c \text{ is assigned to event point } k, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{jk} = \begin{cases} 1, & \text{if job } j \text{ starts at event point } k, \\ 0 & \text{otherwise.} \end{cases}$$

$t_{kc} \geq 0$ is the duration of configuration c in event point k,
$T_k^f \geq 0$ is the completion time of event point k,
$C_j^f \geq 0$ is the completion time of job j.
Then the set of feasible solutions is defined as follows

$$\sum_{c \in C} x_{kc} \leq 1, \ k \in K, \tag{1}$$

$$\sum_{k \in K} \sum_{c \in C} t_{kc} s_{jc} = W_j, \ j \in J, \tag{2}$$

$$\sum_{k \in K} y_{jk} = 1, \ j \in J, \tag{3}$$

$$\sum_{c \in C} x_{kc} q_{jc} - \sum_{c \in C} x_{k-1,c} q_{jc} \leq y_{jk}, \ j \in J, \ k \in K, \tag{4}$$

$$T_k^f \geq T_{k-1}^f + t_{kc}, \ k \in K, \ c \in C, \tag{5}$$

$$t_{kc} \leq x_{kc} T_{max}, k \in K, \ c \in C, \tag{6}$$

$$C_j^f \geq T_k^f - T_{max}(1 - x_{kc}), \ j \in J, \ k \in K, \ c \in C, q_{jc} > 0, \tag{7}$$

$$\sum_{k \in K} \sum_{c \in C} E_c t_{kc} \leq E, \tag{8}$$

$$\sum_{k \in K} (k+1) x_{kc_1} \leq (1 - \sum_{k \in K} x_{kc_1}) k_{max} + \sum_{k \in K} k x_{kc_2}, \tag{9}$$
$$c_1, c_2 \in C, a_{c_1,c_2} > 0.$$

$$x_{k_1,c_1}(k_1 + 1) \leq (x_{k_2,c_2} + (1 - x_{k_2,c_2}) k_{\max}) k_2, \tag{10}$$
$$k_1, k_2 \in K, \ c_1, c_2 \in C, a_{c_1,c_2} > 0,$$

$$t_{kc} \geq 0, \ k \in K, \ c \in C. \tag{11}$$

Inequality (1) indicates that each event point contains at most one configuration. Constraint (2) shows that all jobs are executed in full volumes, while constraints (3)–(4) guarantee that jobs are executed without preemptions. Inequalities (5)–(7) compute completion times for event points and jobs. Constraint (8) gives bound on the energy consumption. Constraints (9) and (10) provide alternative ways to model precedence constraints between configurations.

The makespan is calculated as $\max_{j \in J} C_j^f$ and the total completion time is computed as $\sum_{j \in J} C_j^f$.

Preliminary computations shown that modern solvers within 8 hours and 4 threads did not found even a feasible solution for greater than 10 jobs.

3 Evolutionary Algorithm

We propose an adaptive evolutionary algorithm (EA) with scheme 1. EA models the evolution of a population of individuals that represent tentative solutions of the problem [7]. We encode solutions by permutations of jobs (in the case of several processors jobs are assigned to processors by some prespecified rules together with objective calculation).

Algorithm 1 Elitist Evolutionary Algorithm EA

```
1: Generate $k$ individuals to form the initial population. Keep elites as
   the given number $n_{elites}$ of the best individuals of the initial population
   w.r.t. the objective function.
2: Repeat until the stopping criterion is met.
       2.1 Select individuals from the current population for the reproduction.
       2.2 Apply mutation and recombination to the selected individuals.
       2.3 Build offspring.
       2.4 Update the population and elites.
3: Return the best found individual.
```

Individuals of the initial population are generated randomly or using "arbitrary insertion" method [3], similar to the method for the Traveling Salesman Problem. The last method is applied with the given probability and works according to the rule: some number of jobs are placed in random order, and each subsequent job is chosen randomly and inserted in the position among those already placed where the value of the objective function of the partial solution is minimal.

We use rank or s-tournament selection to choose parent individuals at the reproduction stage. We apply a classic restarting rule: EA is restarted as soon as the current number of objective function calculations becomes equal to the number of objective function calculations at which the best individual was found, plus a given number of objective function calculations n_{FC}. Moreover, our algorithm is elitist and the number of elites n_{elites} is fixed at the preprocessing stage. The solution obtained after reproduction operators replaces the worst individual of the population.

We have a set of crossover operators, and one of them is selected at each iteration using an (ε, δ)-learning scheme. A random crossover operator is selected with probability ε, and the following rule is used with probability $(1-\varepsilon)$. We select an operator that has the best weight with probability δ, and choose a random operator with probabilities proportional to weights otherwise (with probability $(1-\delta)$).

The reward of operator *oper* is calculated as

$$r_{oper} = \begin{cases} w_1, & \text{if the obtained solution becomes a new record,} \\ w_2, & \text{if the obtained solution is a new record in the current restart,} \\ w_3, & \text{if the obtained solution is better than one of the parents,} \\ 0 & \text{otherwise.} \end{cases}$$

The weight of operator *oper* is updated as $R_{oper} = \lambda r_{oper} + (1-\lambda) R_{oper}$.

3.1. Construction of Schedule and Calculation of Objective. Under the given permutation of jobs, the optimal dividing them between the processors is NP-hard problem for makespan criterion even in the case of identical data bus requirements for jobs and two processors (the proof is based on the reduction of the Ordered Partition Problem [5]). Similar question for the total completion time criterion is open. So, we propose a heuristic algorithm to construct a schedule for a permutation.

Energy consumption for jobs is assigned in polynomial time using the convex model on the lower bound of objective [11, 12]. We assume that slowing due to common using of data bus does not change the energy consumption.

We tested two ways to construct a feasible schedule and compute objective for the given permutation of jobs:
1. We take the maximum subset J_0 of the first jobs from the permutation with $\sum_{j \in J_0} size_j \leq m$ and compute data bus consumption for them (see Algorithm 2). Then calculate durations of jobs and identify the shortest job j_s. After the completion of this job we go to the next job j_n from the permutation and try to put it. If $size_{j_n} > size_{j_s}$, then we are waiting for the completion time of the next job in the current schedule. And so on.
2. We take the maximum subset J_0 of the first jobs from the permutation with $\sum_{j \in J_0} size_j \leq m$ and compute data bus consumption for them. Then calculate durations of jobs and identify the shortest job. After the completion of this job we go to the next job or add a *fake* job with zero resource consumption. If the next job requires less percent of data bus that we have, then we try to put this job. Otherwise, we randomly select the delay duration in the range from 0 to the rest duration of the largest job in the current configuration and place fake job of this duration before the next job with the given probability p_{fake}, and add the next job without delay with probability $(1-p_{fake})$. And so on.

The experimental evaluation showed that EA with the second way of objective calculation demonstrates statistically significantly better results than EA with the first way of objective calculation (Wilcoxon test was used). Note that the role of fake jobs is placing idle periods before jobs in order to reduce data bus consumption during these periods.

Algorithm 2 Calculation of data bus consumption for a subset of jobs.

```
1: Put the percentage of the free data bus $freePercent := 100\%$ (the entire
   data bus is free) and set the number of jobs for which the data bus is
   not allocated, $jobsCount$ to be the number of jobs in configuration $k$.
2: While $jobsCount \neq 0$, do:
       2.1 Calculate a percentage of data bus that can be allocated to each
           job: $percent := freePercent/jobsCount$.
       2.2 For jobs $j$ with $b_j < percent$ put: $b'_j := b_j$ and
           $freePercent := freePercent - b_j$,
             $jobsCount := jobsCount - 1$.
       2.3 For remaining jobs $j$ put $b'_j := percent$ and $jobsCount := 0$.
3: Output computed values $b'_j$.
```

Moreover, the permutation is preliminary corrected in accordance with the partial order. We take at each step the next job in the permutation, for which all predecessors are already placed.

3.2. Mutation. The mutation is applied with probability P_m. We apply the classic mutation operators such as swap and shift [7], as well as scramble mutation and heavy-tailed scramble mutation [1].

Scramble mutation operator works as follows. The operator samples a number n_p according to a Poisson distribution with mean $\lambda_p = 1$, selects a random set of n_p elements from $[1 \dots n]$, and applies a random permutation to this set.

We also apply a heavy-tailed scramble mutation operator. We say that an integer random variable X follows a power-law distribution with parameters β and u if

$$Pr[X = i] = \begin{cases} C_{\beta,u} i^{-\beta}, & \text{if } i \in [1 \dots u], \\ 0 & \text{otherwise.} \end{cases}$$

where $C_{\beta,u} = (\sum_{i=1}^{u} i^{-\beta})^{-1}$ denotes the normalization coefficient. We write $X \sim pow(\beta, u)$ and call u the range of X and β the power-law exponent.

The heavy-tailed scramble mutation (with power-law exponent β) is the mutation operator that first samples a number $n_p \sim pow(\beta, n)$, then selects a random subset of n_p elements from $[1 \dots n]$, and finally applies a random permutation on this set.

3.3. Randomized Crossover and Optimal Recombination. The crossover is applied with probability P_c. We operate with randomized and optimized crossover operators. The classic randomized operators are used, such as Order crossover (OX), One point crossover (1PX), Cycle crossover (CX) and Partially mapped crossover (PMX) [9].

The Optimal Recombination Problem (ORP) is formulated as follows. Let us have l parent permutations $\pi^1,\ \pi^2, \dots, \pi^l$. It is required to find a permutation π' such that:
1. $\pi'_i \in \{\pi^1_i, \pi^2_i, \dots, \pi^l_i\}$ for all $i \in \{1, \dots, n\}$;
2. π' has the optimum value of objective function among all permutations that satisfy condition 1).

NP-hardness of the ORP in different cases will be proved by the polynomial reduction of the well-known NP-hard problems: NUMERICAL MATCHING with TARGET SUMS and ORDERED PARTITION.

NUMERICAL MATCHING with TARGET SUMS (strongly NP-hard) [5]: Given two n-element sets $A_1 = \{1, \dots, n\}$ and $A_2 = \{n + 1, \dots, 2n_0\}$, weights of all elements $e_i \in Z^+$, $i = 1, \dots, 2n_0$, and a vector of bounds $(B_1, \dots, B_{n_0})$ with positive integer coordinates. The question is to decide whether a set $A_1 \cup A_2$ can be partitioned into n_0 disjoint subsets $a_1,\ a_2, \dots, a_{n_0}$ such that each set a_j contains exactly one element from the sets A_1, A_2 and $\sum_{i \in a_j} e_i = B_j$.

ORDERED PARTITION (ordinary NP-hard) [5]: Given an ordered set $A = \{1, 2, \dots, 2n_0\}$ and weight e_i of each element $i \in A$ such that $\sum_{i \in A} e_i = 2B$ and $e_i < e_{i+1},\ i = 1, \dots, 2n_0 - 1$. The question is to decide whether A can be partitioned into two subsets A_1 and A_2 so that

$$\sum_{i \in A_1} e_i = \sum_{i \in A_2} e_i = B,$$

and subset A_1 contains only one element from each pair $2i - 1,\ 2i,\ i = 1, \dots, n_0$. Note that $|A_1| = |A_2| = n_0$ in the case of existence.

Theorem 1. *Optimal recombination problem with the makespan criterion and single processor jobs even for the given assignment of jobs to processors in parent solutions is ordinary NP-hard in the cases:*
two parents, two processors and equal resource consumption;
two parents and unit volumes of jobs;
and strongly NP-hard in the case of three processors, and unit volumes of jobs.

Proof. 1), 2) We will use the reduction from the Ordered Partition problem.

1) Put the number of jobs $n = 2n_0$, the number of processors $m = 2$. Jobs correspond to elements of the Ordered Partition problem, processing volume is $W_j = e_j$ and resource consumption $b_j = 50\%$ for job $j = 1, \dots, 2n_0$. Threshold value for the makespan is B and energy budget is $2B$, so, jobs may be executed only with unit speed.

Set the parent permutations for the first processor: $\pi^{11} = (1, 3, \dots, 2n_0 - 1)$, $\pi^{12} = (2, 4, \dots, 2n_0)$ and parent permutations for the second processor: $\pi^{21} = (2, 4, \dots, 2n_0)$, $\pi^{22} = (1, 3, \dots, 2n_0 - 1)$.

Permutation $\pi = (\pi^1, \pi^2)$, for which $\pi^1_i \in \{\pi^{11}_i, \pi^{12}_i\}$, $\pi^2_i \in \{\pi^{21}_i, \pi^{22}_i\}$, $i = 1, \ldots, n_0$, and makespan is bounded by B, exists if and only if the Ordered Partition Problem has a positive answer.

2) Put the number of jobs $n = 2n_0$, the number of processors $m = n_0$. Jobs correspond to elements of the Ordered Partition problem, processing volume is $W_j = 1$ and resource consumption is $b_j = \frac{100\times e_j}{B}\%$ for job $j = 1, \ldots, 2n_0$. Threshold value for the makespan is 2 and energy budget is $2B$, so, jobs may be executed only with unit speed.

Set the parent permutations for the j-th processor: $\pi^{j1} = (2j-1, 2j)$, $\pi^{j2} = (2j, 2j-1)$, $j = 1, \ldots, n_0$.

Permutation $\pi = (\pi^1, \pi^2, \ldots, \pi^{n_0})$, for which $\pi^j_i \in \{\pi^{j1}_i, \pi^{j2}_i\}$, $i = 1, 2$, $j = 1, \ldots, n_0$, and makespan is bounded by 2, exists if and only if the Ordered Partition Problem has a positive answer. Here we want to divide the set of jobs into two parts of equal total data bus consumption if it is possible.

3) We will use the reduction from the Numerical Matching with Target Sums problem. Put the number of jobs $n = 3n_0$, the number of processors $m = 3$. The first $2n_0$ jobs correspond to elements of the Numerical Matching with Target Sums problem, processing volume is $W_j = 1$ and resource consumption is $b_j = \frac{100\times e_j}{M}$ for job $j = 1, \ldots, 2n_0$. Jobs $2n_0+1, \ldots, 3n_0$ have unit volumes and resource consumption $b_j = \frac{100\times(M-B_j)}{M}$ for job $j = 2n_0+1, \ldots, 3n_0$. Here M is the sufficiently big constant. Threshold value for the makespan is n_0 and energy budget is $3n_0$, so, jobs may be executed only with unit speed.

Put the i-th parent permutation for processors:
$\pi^{1i} = (i, i+1 \bmod n_0, \ldots, i+n_0-1 \bmod n_0)$,
$\pi^{2i} = (n_0+i, n_0+i+1 \bmod 2n_0, \ldots, n_0+i+n_0-1 \bmod 2n_0)$,
$\pi^{3i} = (2n_0+1, \ldots, 3n_0)$, $i = 1, \ldots, n_0$.

Permutation $\pi = (\pi^1, \pi^2, \pi^3)$, for which $\pi^j_i \in \{\pi^{jl}_i\}^{n_0}_{l=1}$, $i = 1, 2, \ldots, n_0$, $j = 1, 2, 3$, and makespan is bounded by n_0, exists if and only if the Numerical Matching with Target Sums problem has a positive answer. Here we try to reorder jobs on the first two processors such that the data bus consumption is equal to $\frac{100\times B_t}{M}$ for unit intervals $t = 1, \ldots, n_0$ if possible. □

Theorem 2. *Optimal recombination problem with the total completion time criterion even for the given assignment of jobs to processors in parent solutions is ordinary NP-hard in the cases:*
at least one two-processor job, two parents, two processors and equal resource consumption for single-processor jobs;
two parents and unit durations of jobs;
and strongly NP-hard in the case of three processors, and unit durations of jobs.

Proof. We show the NP-hardness for the case, when energy resource is not considered. The provided reductions can be generalized to the case of energy resource using approaches from papers [21] (see Theorem 1) and [12] (see Theorem 3.1).

1), 2) We will use the reduction from the Ordered Partition problem. This reduction is similar to the case of makespan criterion, but has unique properties due to the objective.

1) Put the number of jobs $n = 2n_0 + 1$, the number of processors $m = 2$. We have $2n_0$ single-processor jobs and one two-processor job. Single-processor jobs correspond to elements of the Ordered Partition problem, processing volume $W_j = e_j$ and resource consumption $b_j = 50\%$ for job $j = 1, \dots, 2n_0$. Two-processor job is characterized by volume $W_{2n_0+1} = M > \sum_{i=1}^{n_0}(n_0+i+1)(e_{2i-1}+e_{2i}) + \frac{1}{2}\sum_{i=1}^{n_0}(e_{2i-1}+e_{2i})$ and resource consumption $b_{2n_0+1} = 100\%$. Threshold value for the total completion time is $L := \sum_{i=1}^{n_0}(n_0+i+1)(e_{2i-1}+e_{2i}) + \frac{1}{2}\sum_{i=1}^{n_0}(e_{2i-1}+e_{2i}) + M$, so, jobs may be executed only with unit speed.

We set parent permutations for the first processor: $\pi^{11} = (1, 3, \dots, 2n_0 - 1, 2n_0 + 1)$, $\pi^{12} = (2, 4, \dots, 2n_0, 2n_0 + 1)$, and parent permutations for the second processor: $\pi^{21} = (2, 4, \dots, 2n_0, 2n_0+1)$, $\pi^{22} = (1, 3, \dots, 2n_0-1, 2n_0+1)$.

Permutation $\pi = (\pi^1, \pi^2)$, for which $\pi^1_i \in \{\pi^{11}_i, \pi^{12}_i\}$, $\pi^2_i \in \{\pi^{21}_i, \pi^{22}_i\}$, $i = 1, \dots, n_0 + 1$, and the total completion time is bounded by L, exists if and only if the Ordered Partition Problem has a positive answer. We want to divide single-processor jobs between two processors such that the two-processor job starts at time $\frac{1}{2}\sum_{i=1}^{n_0}(e_{2i-1}+e_{2i})$ if it is possible.

2) Put the number of jobs $n = 2n_0$, the number of processors $m = n_0$. Jobs are single-processor and correspond to elements of the Ordered Partition problem, processing volume is $W_j = 1$ and resource consumption $b_j = \frac{100\times e_j}{B}\%$ for job $j = 1, \dots, 2n_0$. Threshold value for the total completion time is $3n_0$, so, jobs may be executed only with unit speed.

We set parent permutations for the i-th processor as $\pi^{i1} = (2i-1, 2i)$, $\pi^{i2} = (2i, 2i-1)$, $i = 1, \dots, n_o$ and prove NP-hardness similarly to the makespan criterion.

3) We will use the reduction from the Numerical Matching with Target Sums problem. Put the number of jobs $n = 3n_0$, the number of processors $m = 3$. The first $2n_0$ jobs correspond to elements, processing volume $W_j = 1$, $size_j = 1$ and resource consumption $b_j = \frac{100\times e_j}{M}$ for job $j = 1, \dots, 2n_0$. Jobs $2n_0 + 1, \dots, 3n_0$ have unit volumes, $size_j = 1$ and resource consumption $b_j = \frac{100\times(M-B_j)}{M}$ for job $j = 2n_0 + 1, \dots, 3n_0$. Threshold value for the total completion time criterion is $3\sum_{i=1}^{n_0} i = 3\frac{n(n+1)}{2}$, so, jobs may be executed only with unit speed.

Then we put parent permutations as for the makespan criterion and continue the proof similarly. $\square$

Now we go to discussion of the solving methods for the ORP. Construct a bipartite graph $G = (J_n, J, U)$ with two parts of the vertices set J_n, J of equal

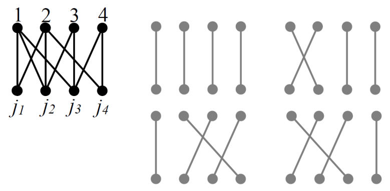

FIG. 1. Graph $G = (X_4, X, U)$ for parent permutations $\pi^1 = (j_3,\ j_1,\ j_2,\ j_4)$, $\pi^2 = (j_2,\ j_2,\ j_3,\ j_4)$, $\pi^3 = (j_1,\ j_4,\ j_2,\ j_3)$, $\pi^4 = (j_2,\ j_1,\ j_3,\ j_4)$ and possible perfect matchings.

sizes and the set of edges $U = \{\{i, x\} : i \in J_n,\ x \in \{\pi_i^1, \pi_i^2, \ldots, \pi_i^l\}\}$. The left part of the vertex set represents positions and the right part shows jobs. By construction we see that there is a one-to-one correspondence between perfect matchings of G and feasible permutations (Fig. 1).

A feasible schedule of the ORP can be found in polynomial time by solving the well known assignment problem [17], and the optimal solution may be computed in $O(n^{2.5} + n^2 n_{pm})$ time by enumerating perfect matchings (see, e.g., [19]) in the corresponding bipartite graph G (n_{pm} is the number of perfect matchings) and computing objectives for appropriate schedules. Here time $O(n^{2.5})$ corresponds to compute one perfect matching and time $O(n^2 n_{pm})$ is used for going from the current perfect matching to the next one and calculating the required metrics.

In the case of two parents we have the following optimistic property: “Almost all” pairs of parent solutions give the ORP with at most n feasible solutions, that can be found in polynomial time [2]. We also consider partially optimized crossover, where at most the given number of feasible solutions (the given number of perfect matchings) is considered in the solving method of the ORP. This number depends on the problem dimension (the number of jobs).

The ORP can be formulated in terms of MIP subproblem, when we fix Boolean variables with identical values for parents and solve the subproblem on the rest variables. The previous research shown that the approach based on enumerating perfect matchings demonstrates better results than MIP approach for scheduling problems on permutations (see, e.g., [21, 22]).

4 Convergence of the Evolutionary Algorithm

Now we analyze the convergence to the optimum of the proposed algorithm and estimate the expected number of iterations in the suggestion that the energy-resource is not considered.

Theorem 3. *The expected number of iterations of EA is bounded by*

$$O\left((n-1)\left(\frac{k^2n^2p_{\max}}{P_m(1-P_c)\Delta_p p_{fake}^{\min}}\right)^{n-1}\right),$$

if the rank selection is chosen as the selection operator. Here n is the total number of jobs, k is the size of the population, $p_{fake}^{\min} = \min\{p_{fake}, 1-p_{fake}\}$, $p_{\max}$ is duration of the longest jobs with slowing effect due to common using of data bus, Δ_p is the duration portion, which is a multiple of all job durations taking into account possible slowing effect.

Proof. We denote by π^* the permutation, corresponding to the starting times of jobs in optimal solution. Now we consider an arbitrary permutation π^0 from the initial population. Let d^0 be the distance (the number of positions with different values) between π^0 and π^*. We will calculate upper bound on the expected number of iterations required to obtain π^* from π^0.

We will go from π^0 to π^* in at most (d^0-1) iterations only by means of mutation, where the distance between the selected permutation and the optimal one is decreased at each iteration. For the given pair of solutions a crossover operator is not applied with probability $(1-P_c)$, and for the given solution a mutation operator is applied with probability P_m.

We will consider such transition from π^0 to π^*, that at least one job is placed in the same position as in π^* by the mutation at each iteration and the obtained permutation is selected at the next iteration.

The probability of an individual being selected in the rank selection operator is $\frac{2}{k(k+1)}$. And the probability that one of the jobs will be in the same position as in the optimal permutation after applying swap or insert mutation has a lower bound $\frac{1}{n^2}$. Thus, the probability of obtaining π^* from π^0 in at most d^0-1 iterations is bounded by $\left(\frac{2}{k(k+1)}P_m\frac{1}{n^2}(1-P_c)\right)^{d^0-1}$. In construction of a schedule for permutation π^* the fake jobs of necessary durations are placed before real jobs with probability no less $\left(\frac{\Delta_p p_{fake}^{\min}}{p_{\max}}\right)^{n-1}$. Here we suppose that the duration of a fake job is set as $k\Delta_p$, and k is selected randomly from $\{1,\dots,\frac{p_{rest}}{\Delta_p}\}$, where p_{rest} is the rest duration of the largest job in the corresponding configuration (see Section 3.1).

Denote the estimation on the probability of constructing an optimal solution in at most d^0-1 iterations as

$$p^* := \left(\frac{2}{k(k+1)}P_m\frac{1}{n^2}(1-P_c)\right)^{d^0-1}\cdot\left(\frac{\Delta_p p_{fake}^{\min}}{p_{\max}}\right)^{n-1}.$$

Now we consider a series of (d^0-1) iterations. Let S_i be an event indicating that an optimal solution is absent in the population of series i. Then the probability $P(S_i) \le 1-p^*$ for any prehistory, therefore, $P(S_1\&\dots\&S_t) \le (1-p^*)^t$.

Let us denote by Ξ the random variable equal to the number of the first series, when an optimal solution is obtained. We have

$$E[\Xi] = \sum_{i=0}^{\infty} P(\Xi > i) = 1 + \sum_{i=0}^{\infty} P(S_1 \& \ldots \& S_i) \le 1 + \sum_{i=0}^{\infty} (1-p^*)^i = \frac{1}{p^*}.$$

Thus, the expected number of iterations does not exceed:

$$T := (d^0 - 1)\left(\frac{k^2 n^2}{P_m(1-P_c)}\right)^{d^0-1}\left(\frac{p_{\max}}{\Delta_p p_{fake}^{\min}}\right)^{n-1}.$$

Let us denote by Θ the random variable equal to the number of iterations until the optimal solution is obtained, by $\pi^{1,0}$ the first individual of the initial population Π^0. Then, according to the formula of the total probability for conditional mathematical expectations, we have:

$$E[\Theta] = \sum_{\pi \in \Pi^0} E[\Theta | \pi^{1,0} = \pi] \cdot P\{\pi^{1,0} = \pi\} \le T \sum_{\pi \in \Pi^0} P\{\pi^{1,0} = \pi\} = T.$$

The statement follows from inequality $d^0 \le n$ and the property that after the optimal solution is reached, such solution will be kept in each subsequent population. □

Similar theorem is also valid when we use tournament selection. The probability that the required permutation will be selected as a parent by the s-tournament selection is no less than $\frac{1}{k^s}$.

Theorem 4. *The expected number of iterations of EA is bounded by*

$$O\left((n-1)\left(\frac{k^s n^2 p_{\max}}{P_m(1-P_c)\Delta_p p_{fake}^{\min}}\right)^{n-1}\right),$$

if s-tournament selection is chosen as the selection operator.

Theorems 3 and 4 guarantee that our EA almost surely converges to an optimum as the number of iterations tends to infinity.

5 Computational Experiment

The experiment was carried out on a computer with 16 GB of RAM and Intel Core i7-8565U 1.80 GHz CPU. For the computational experiment, the following procedures from the Intel MKL (Math Kernel Library) are used as jobs:

– copying a vector to another vector,
– calculation of the sum of magnitudes of the vector elements,
– calculation of a vector-scalar product and adding the result to a vector,
– calculation of the QR factorization of a matrix.

The choice of such procedures is due to the fact that they consume the data bus in different ways. The input data to the procedures has different sizes (for procedures with vectors, this is the vector length, for procedures with matrices, this is the matrix size). For vectors, the dimensions from 10 to 70 million elements were used, for matrices, the dimensions varied from 1000 to 1300. Data bus consumption for jobs is calculated by the method presented in [4]. Parameter $\alpha := 2$ for calculating energy consumption.

Input parameters of the instances:

– The number of jobs $n = 50,\ 100$.
– Precedence constraints: jobs without partial order and binary tree.
– The number of processors (cores) $m = 12,\ 24$.
– Data bus consumption $b_j \in \{4, 7, 8, 9, 10, 11\}$, in %.

5.1. Testing Evolutionary Algorithm. We carried out the experiment on EA with scramble mutation (SM) and heavy-tailed scramble mutation (H-SM) on series constructed for 50 and 100 jobs with bitree partial order (B-PO) and without partial order (N-PO).

The total budget on the objective function calculations is 1.5 millions, the number of calculations before the next restart n_{FC} is 200 thousands. We put weights in the adaptive scheme $w_1 = 1,\ w_2 = 0.8$ based on the preliminary computational experiment. Numerical parameters of EA are tuned at the preprocessing stage by IRACE package (see Table 1). Here values were configured by testing on instances with 50 jobs. For all experiments the number of parents in the optimized crossover was set to two and the maximum number of the considered permutations was set to $0.5\log n$ for the partially optimized crossover.

The lower bounds on objectives and energy consumption for jobs are calculated using the convex models for both criteria, taking into account only energy resource and ignoring slowing due to data bus bandwidth [11, 12]. The objective function deviations of solutions obtained by EA from the lower bound (LB) and from the objective of solutions found by the greedy algorithm (GR) [4] are presented in tables 2, 3, 4 and 5.

param	$C_{\max}$				$\sum_j C_j$			
	SM, N-PO	H-SM, N-PO	H-SM, B-PO	SM, B-PO	SM, N-PO	SM, B-PO	H-SM, B-PO	H-SM, N-PO
k	232	246	200	238	244	228	198	202
n_{elites}	192	74	2	232	28	114	22	80
select	s = 14	rank	rank	rank	rank	rank	rank	rank
p_{fake}	0.88	0.52	0.5	0.87	0.74	0.48	0.22	0.7
β	-	1.44	1.5	-	-	-	1.17	1.16
λ	0.44	0.54	0.2	0.26	0.64	0.81	0.64	0.13
w_3	0.12	0.21	0.3	0.29	0.51	0.53	0.69	0.65
ε	0.064	0.03	0.01	0.065	0.061	0.054	0.026	0.04
δ	0.08	0.085	0.01	0.042	0.082	0.08	0.076	0.011

TABLE 1. Values of parameters found by IRACE package.

	50				**100**			
	SM		**H-SM**		**SM**		**H-SM**	
	LB	**GR**	**LB**	**GR**	**LB**	**GR**	**LB**	**GR**
min	0.0	0.01	0.0	0.01	58.53	0.0	52.87	0.0
avg	0.9	12.71	0.84	12.79	65.18	0.1	59.32	3.86
max	1.23	24.27	1.19	24.57	76.06	5.2	65.96	12.17

Table 2. Results for the jobs without partial order and $C_{\max}$ criterion: relative deviation from the lower bound and from the solution of greedy algorithm [4], %.

	50				**100**			
	SM		**H-SM**		**SM**		**H-SM**	
	LB	**GR**	**LB**	**GR**	**LB**	**GR**	**LB**	**GR**
min	11.55	0.0	11.55	0.0	78.03	0.0	58.65	0.0
avg	74.62	4.05	74.62	4.05	118.57	3.3	100.93	13.12
max	228.9	35.53	228.9	35.53	207.93	16.74	207.93	22.94

Table 3. Results for the jobs with bitree partial order and $C_{\max}$ criterion: relative deviation from the lower bound and from the solution of greedy algorithm [4], %.

	50				**100**			
	SM		**H-SM**		**SM**		**H-SM**	
	LB	**GR**	**LB**	**GR**	**LB**	**GR**	**LB**	**GR**
min	50.05	17.61	50.13	17.61	121.01	39.56	114.78	42.31
avg	62.0	27.26	62.0	27.26	139.32	51.54	136.85	53.91
max	74.04	34.31	74.04	34.31	154.97	68.27	155.28	72.0

Table 4. Results for the jobs without partial order and $\sum_j C_j$ criterion: relative deviation from the lower bound and from the solution of greedy algorithm [4], %.

	50				**100**			
	SM		**H-SM**		**SM**		**H-SM**	
	LB	**GR**	**LB**	**GR**	**LB**	**GR**	**LB**	**GR**
min	90.61	0.0	90.61	0.0	162.88	0.0	157.69	0.3
avg	184.8	2.79	184.8	2.79	242.68	4.76	234.23	7.57
max	533.67	14.48	533.67	14.48	515.63	13.82	505.75	16.05

Table 5. Results for the jobs with bitree partial order and $\sum_j C_j$ criterion: relative deviation from the lower bound and from the solution of greedy algorithm [4], %.

As we can see, EA with SM and EA with H-SM demonstrate similar good results on instances with 50 jobs due to the big number of calculations for this dimension. EA with H-SM demonstrates sufficiently better results in comparison with EA with SM and greedy algorithm in the case of 100 jobs. The statistical analysis of experimental data using the Wilcoxon test shows that difference between the record values of EA with H-SM and EA with SM is statistically significant at level less than 0.05 on all series with 100 jobs. For the total completion time criterion EA shows more sufficient improvement of results of greedy algorithm on instances without partial order in comparison with instances with partial order. Deviation from the lower bound is better for the makespan criterion than for the total completion time criterion.

The results of (ε, δ)-learning scheme show that CX, 1PX and partially optimized crossover demonstrate leading positions during EA iterations. CX is the more effective randomized operator for instances without partial order and 1PX is the more effective randomized operator for instances with bitree order. Significant differences in the work of operators for criteria $\sum_j C_j$ and $C_{\max}$ are not identified. We also tested our EA based on adaptive scheme without optimized crossover operators on instances without partial order. The results are statistically significantly worse than the results with optimized operators.

5.2. Comparison with Solver. Here we present the results of evolutionary algorithm EA with heavy-tailed scramble mutation H-SM for small-size instances ($n = 4,\ 6,\ 7,\ 8,\ 10$), and compare them to the results of the Gurobi solver reached withing 20 minutes by model (1)–(11). Four series of are considered: instances without partial order (no order), instances with bitree partial order (bitree), instances with one-to-many-to-one partial order (otmto) and instances with random partial order (random), see [4]. Each series contains 192 instances.

	no order	**bitree**	**otmto**	**random**
min	0.04	0.00	0.00	0.01
avg	1.47	1.82	0.97	1.94
max	19.89	20.53	19.20	17.63

Table 6. Relative deviation of GA solution from Gurobi solution for 4 jobs, %.

	no order	**bitree**	**otmto**	**random**
min	0.07	0.09	-0.77	-0.12
avg	0.15	0.83	3.30	1.99
max	0.29	9.89	26.98	13.88

Table 7. Relative deviation of GA solution from Gurobi solution for 6 jobs, %.

	no order	**bitree**	**otmto**	**random**
min	-3.88	-0.11	-0.02	0.00
avg	0.10	0.37	2.36	1.82
max	0.49	4.73	25.19	14.47

Table 8. Relative deviation of GA solution from Gurobi solution for 7 jobs, %.

	no order	**bitree**	**otmto**	**random**
min	-2.60	-1.82	-2.33	-2.54
avg	-0.06	0.40	1.41	1.72
max	0.37	10.41	20.96	13.78

Table 9. Relative deviation of GA solution from Gurobi solution for 8 jobs, %.

	no order	**bitree**	**otmto**	**random**
min	-10.20	-12.74	-10.90	0.00
avg	-0.51	-1.45	0.04	1.22
max	0.36	0.36	5.84	8.75

Table 10. Relative deviation of GA solution from Gurobi solution for 10 jobs, %.

Note that Gurobi solver was not found an optimal solution in all instances (6 jobs: 7 instances, 7 jobs: 31 instances; 8 jobs: 73 instances; 10 jobs: 115 instances). EA demonstrates no more than 3.5% deviation from the Gurobi solution on average and it shows an advantage as dimension increased.

Conclusion

We proposed a new adaptive evolutionary algorithm with optimized crossover and scramble mutation for the multi-processor scheduling with resources. The problem specific decoding scheme is used allowing to provide assignment of processors and resources. The complexity of operators and convergence of the algorithm were theoretically analyzed. The experimental evaluation showed that the developed evolutionary algorithm demonstrated competitive results on instances generated based on procedures from the Intel MKL library.

References

[1] B. Doerr, Y. Ghannane, M. Ibn Brahim, *Runtime analysis for permutation-based evolutionary algorithms*, Algorithmica, **86**:1 (2024), 90–129. Zbl 7785276

[2] A. Eremeev, Y. Kovalenko, *Optimal recombination in genetic algorithms for combinatorial optimization problems. II*, Yugoslav. J. Oper. Res., **24**:2 (2014), 165–186. Zbl 1458.90645

[3] A.V. Eremeev, Y.V. Kovalenko, *A memetic algorithm with optimal recombination for the asymmetric travelling salesman problem*, Memetic Comp., **12** (2020), 23–36.

[4] A.V. Eremeev, A.A. Malakhov, M.A. Sakhno, M.Y. Sosnovskaya, *Multi-core processor scheduling with respect to data bus bandwidth*, In Olenev, Nicholas (ed.) et al., *Advances in optimization and applications. 11th international conference, OPTIMA 2020*, Commun. Comput. Inf. Sci., **1340**, Springer, Cham, 2020, 55–69. Zbl 1533.90053

[5] M.R. Garey, D.S. Johnson, *Computers and intractability*, Freeman, San Francisco, 1979. Zbl 0411.68039

[6] M.E.T. Gerards, J.L. Hurink, P.K.F. Holzenspies, *A survey of offline algorithms for energy minimization under deadline constraints*, J. Sched., **19**:1 (2016), 3–19. Zbl 1341.90046

[7] J.H. Holland, *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence*, The MIT Press, Ann Arbor, 1992. MR0441393

[8] J. Józefowska, J. Weglarz, *Scheduling with resource constraints – continuous resources*, in: J.Y-T. Leung (ed.), *Handbook of scheduling: Algorithms, models, and performance analysis*, Chapman&Hall/CRC, 2004, 24-1–24-15.

[9] T. Kellegöz, B. Toklu, J. Wilson, *Comparing efficiencies of genetic crossover operators for one machine total weighted tardiness problem*, Appl. Math. Comput., **199**:2 (2008), 590–598. Zbl 1140.90402

[10] F. Kong, N. Guan, Q. Deng, W. Yi, *Energy-efficient scheduling for parallel real-time tasks based on level-packing*, Proceedings of the 2011 ACM Symposium on Applied Computing, ACM, New York, 2011, 635–640.

[11] A. Kononov, Y. Zakharova, *Speed scaling scheduling of multiprocessor jobs with energy constraint and makespan criterion*, J. Glob. Optim., **83**:3 (2022), 539–564. Zbl 1495.90076

[12] A.V. Kononov, Y.V. Zakharova, *Speed scaling scheduling of multiprocessor jobs with energy constraint and total completion time criterion*, Int. J. Artif. Intell., **21**:2 (2023), 109–129.

[13] J.K. Lenstra, A.H.G. Rinnooy Kan, P. Brucker, *Complexity of machine scheduling problems*, Ann. of Discrete Math. **1**, 343–362 (1977). Zbl 0353.68067

[14] J.K. Lenstra, A.H.G. Rinnooy Kan, Complexity of scheduling under precedence constraints. Oper. Res., **26**:1 (1978), 22–35. Zbl 0371.90060

[15] K. Li, *Energy efficient scheduling of parallel tasks on multiprocessor computers*, J. Supercomput., **60** (2012), 223–247.

[16] A. Merkel, J. Stoess, F. Bellosa, *Resource-conscious scheduling for energy efficiency on multicore processors*, Proc. 5th European Conference on Computer Systems (EuroSys'10), Paris, France, ACM, New York, 2010, 153–166.

[17] J. Munkres, *Algorithms for the assignment and transportation problems*, J. Soc. Ind. Appl. Math., **5**:1 (1957), 32–38. Zbl 0083.15302

[18] K.J. Nesbit, N. Aggarwal, J. Laudon, J.E. Smith, *Fair Queuing Memory Systems*, In Proc. of 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006, 208–222.

[19] Uno, T., *Algorithms for enumerating all perfect, maximum and maximal matchings in bipartite graphs*, In Leong, Hon Wai (ed.) et al., *Algorithms and computation*, 8th

International Symposium, ISAAC'97, Proceedings, Lect. Notes Comput. Sci., **1350**, Springer, Berlin, Heidelberg, 1997, 92–101. Zbl 0897.05066

[20] Y.V. Zakharova, M.Y. Sakhno, *Adaptive genetic algorithm with optimized operators for scheduling in computer systems*, In Shi, Z., Torresen, J., Yang, S. (eds), *Intelligent Information Processing XII. IIP 2024*, IFIP Advances in Information and Communication Technology, **703**, 2024, Springer, Cham, 317–328.

[21] Y. Zakharova, M. Sakhno, Complexity and heuristic algorithms for speed scaling scheduling of parallel jobs with energy constraint. J. Comput. Appl. Math. A **457** (2025), Article ID 116254. Zbl 1560.90068

[22] Y.V. Zakharova, A.O. Zakharov, *Integer programming models and metaheuristics for customer order scheduling*, In Eremeev, Anton (ed.) et al., *Mathematical optimization theory and operations research: recent trends. MOTOR 2024*, CCIS, **2239**, Springer, Cham, 2024, 276–290. Zbl 8077309

[23] Y.V. Zakharova, *Population local search for single processor energy efficient scheduling problem*, In Sergeyev, Y.D., Kvasov, D.E., Astorino, A. (eds), *Numerical computations: theory and algorithms. NUMTA 2023*, Lecture Notes in Computer Science, **14476**, Springer, Cham, 2025, 400–408.

[24] S. Zhuravlev, S. Blagodurov, A. Fedorova, *Addressing shared resource contention in multicore processors via scheduling*, In: Proc. 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS'10), Pittsburgh, USA. ACM, New York, 2010, 129–142.

Yulia Victorovna Zakharova
Novosibirsk State University,
str. Pirogova, 1
630090, Novosibirsk, Russia
Dostoevsky Omsk State University,
str. Neftezavodskaya, 11
644077, Omsk, Russia
Email address: yzakharova@ofim.oscsbras.ru

Maria Yurievna Sakhno
Sobolev Institute of Mathematics, Omsk Department
str. Pevtsova, 13
644043, Omsk, Russia
Email address: sosnovskayamy@gmail.com