Evolutionary Algorithms for Customer Order Scheduling

P. Borisovsky, Yu. Zakharova, A. Zakharov

Sobolev Institute of Mathematics SB RAS

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

Problem Statement (one machine)

Input data

n is the number of customers
m is the number of products
$p_{ij} \geq 0$ is the duration of producing product j for customer i
$s_{jj'} \geq 0$ is the setup time from product j to product j'
s'_j is the initial setup to product j

Solution representation

We define operation as a pair of customer and product type (i, j), $i = 1, \ldots, n$, $j = 1, \ldots, m$. Operations should be scheduled without preemptions. A set of feasible solution Π consists of permutations of operations.

Problem Statement (multiple machines)

Input data

n is the number of customers
m is the number of products (machines)
each machine produces a unique product
$p_{ij} \geq 0$ is the duration of producing product j for customer i

Solution representation

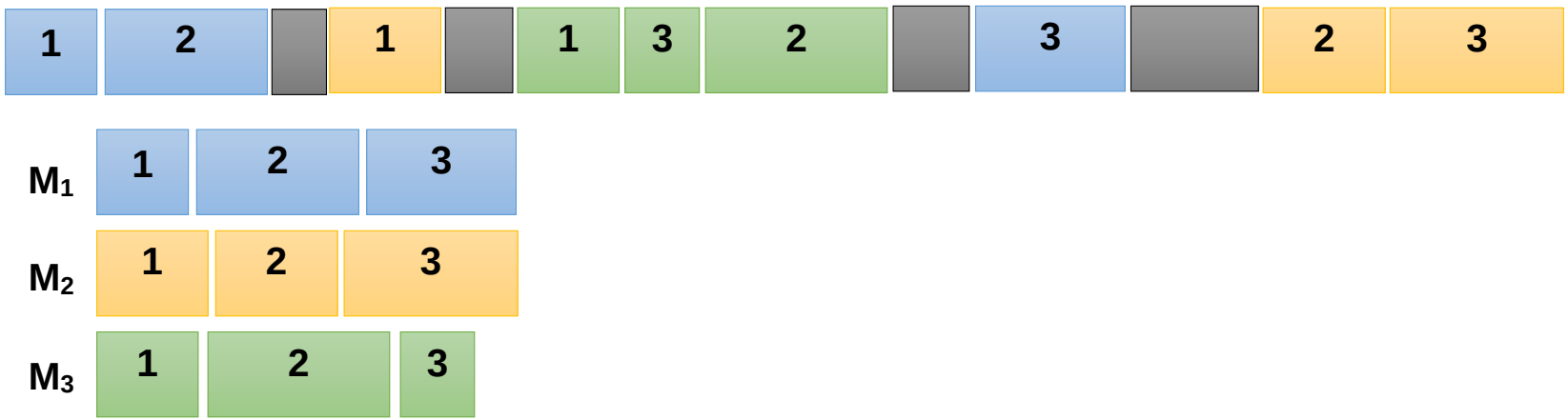
Solutions are encoded by permutations of orders. Products should be scheduled without preemptions. A set of feasible solution Π consists of permutations of orders.

Criterion

$\sum_{i=1}^{n} C_i$ is the total completion time.

The single machine problem is NP-hard even in the case when setup times are sequence independent.

The multi machine problem is NP-hard even in the case of two machines.



Previous Research (single machine)

Hazir, O., Gunalay, Y., Erel, E. (2008). Customer order scheduling problem: a comparative metaheuristics study. Inter. Jour. Adv. Manuf. Techn.

Erel, E., Ghosh, J. B. (2007). Customer order scheduling on a single machine with family setup times: Complexity and algorithms. Appl. Math. Comp.

Cetinkaya, F. C., Yeloglu, P., Catmakas, H. A. (2021). Customer order scheduling with job-based processing on a single-machine to minimize the total completion time. Inter. Jour. Indust. Engin. Comput.

Wu, Ch.-Ch., Bai, D., Zhang, X. et al (2021). A robust customer order scheduling problem along with scenario-dependent component processing times and due dates. Jour. Manuf. Syst.

Previous Research (multiple machines)

Wang, G., Cheng, T. C. E. (2007). Customer order scheduling to minimize total weighted completion time. Omega.

Framinan, J. M., Perez-Gonzalez, P. (2017). New approximate algorithms for the customer order scheduling problem with total completion time objective. Comp. Oper. Res.

Borisovsky, P., Eremeev, A., Kallrath J. (2020). Multi-product continuous plant scheduling: combination of decomposition, genetic algorithm, and constructive heuristic. Inter. Jour. of Prod. Res.

Shi Zh., Ma H., Ren M., Wu T., Yu A.J. (2021). A learning-based two-stage optimization method for customer order scheduling. Comput. Oper. Res.

Genetic Algorithm (GA)

Scheme

Generate tentative solutions to form the initial population.
Repeat until the stopping criterion is met.

2.1: Select two solutions from the current population.

2.2: Build offspring by a recombination (crossover) and a mutation.

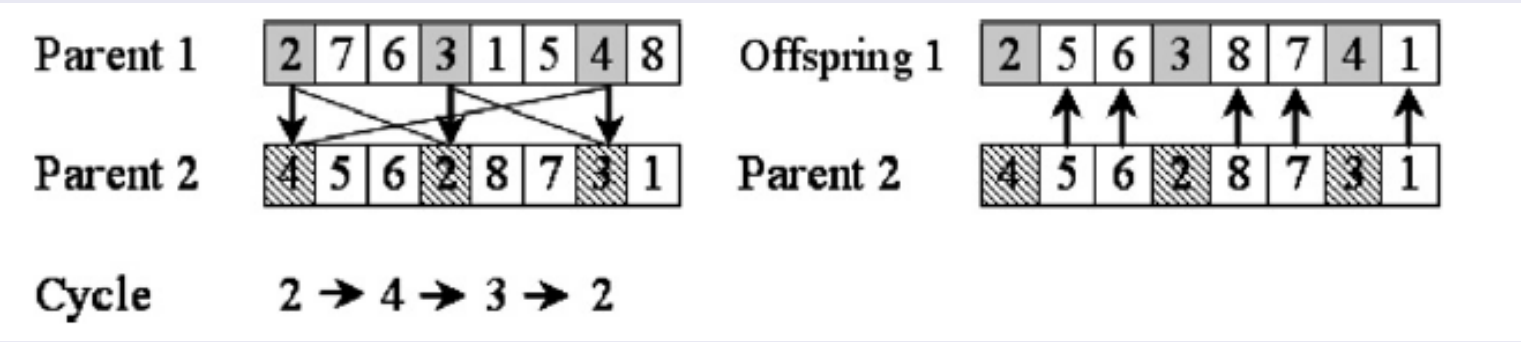
2.3: Choose solutions for the next generation.

Return the best found solution.

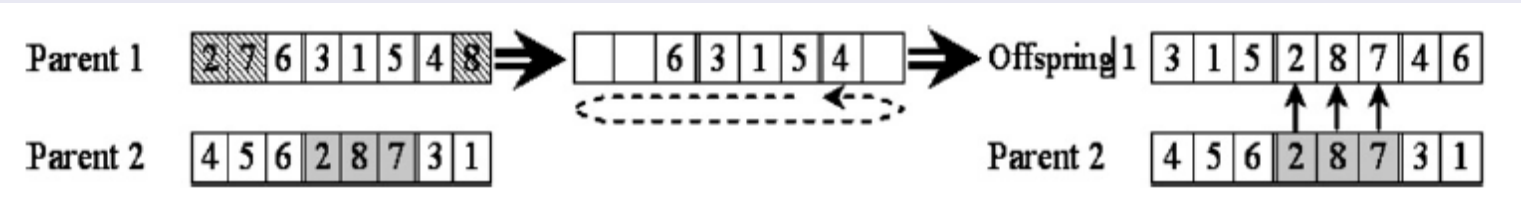
Convergence of GA

GA algorithm almost surely converges to an optimum as the number of iterations tends to infinity.

Cycle Crossover (CX)

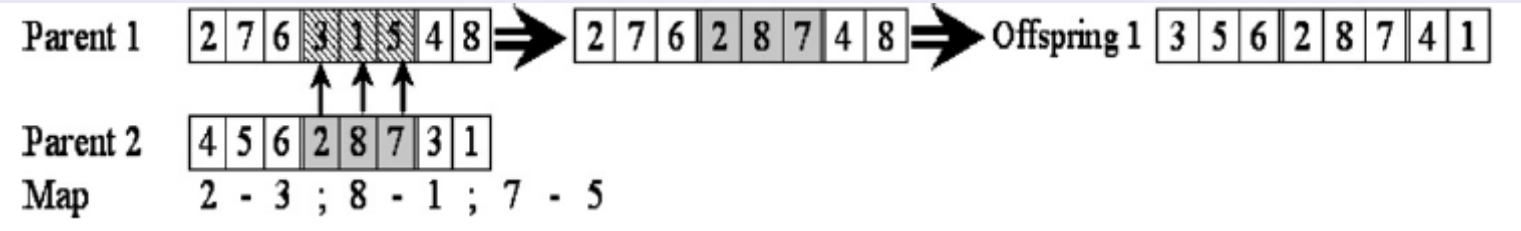


Order Crossover (OX)

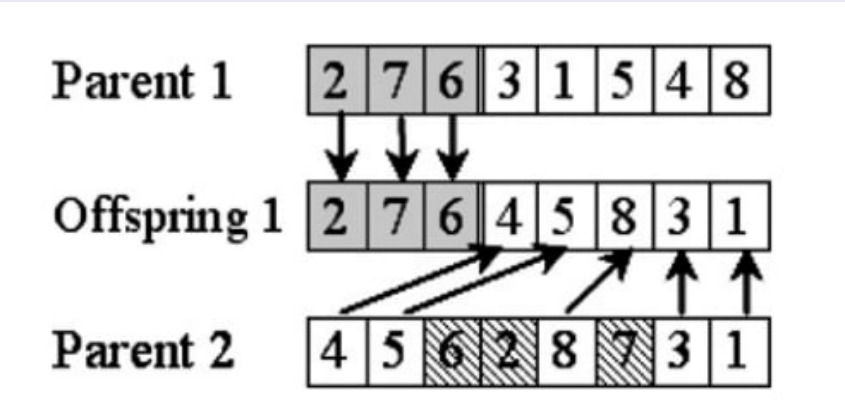


Crossover Operators

Partially Mapped Crossover (PMX)

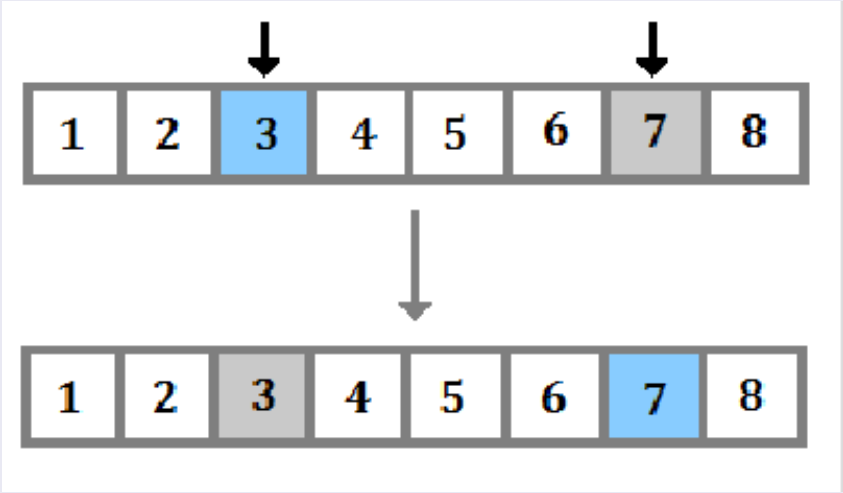


One Point Crossover (1PX)

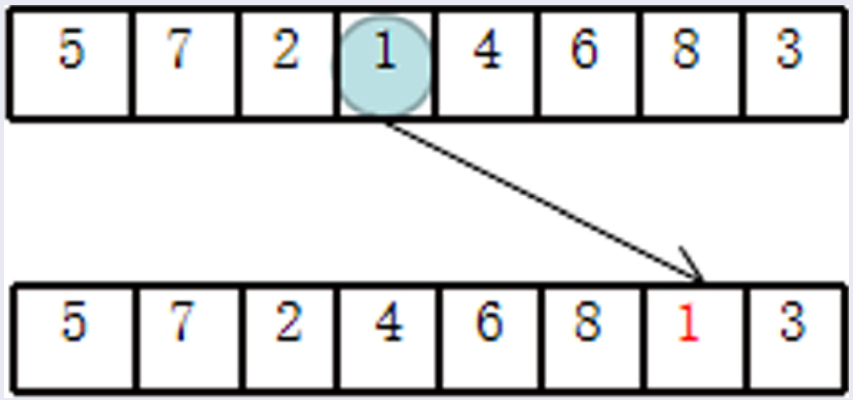


Mutation Operators

Exchange (swap) mutation



Shift (insert) mutation



Optimal Recombination

We have two parent permutations π^1 and π^2 . It is required to find a permutation π' such that:

(I) $\pi'_i = \pi^1_i$ or $\pi'_i = \pi^2_i$ for all $i = 1, \ldots, k$;

(II) π' has the optimum value of objective function among all permutations that satisfy condition (I).

The complexity status of the ORP for the total completion time criterion is an open question.

Solving Approach

Main Idea

- $\bar{G} = (X_k, X, \bar{U})$ is the bipartite graph.
- $\bar{U} = \{\{i, x\} : i \in X_k,\ x \in X^i\}$ is the set of edges.
- Vertices of the left part $\leftrightarrow$ positions.
- Vertices of the right part $\leftrightarrow$ jobs.
- There is a one-to-one correspondence between the set of perfect matchings $\mathcal{W}$ in the graph $\bar{G}$ and the set Π of feasible permutations to a problem instance [a].

[a] Serdyukov A.I. (1978); Eremeev A., Kovalenko Yu. (2017)

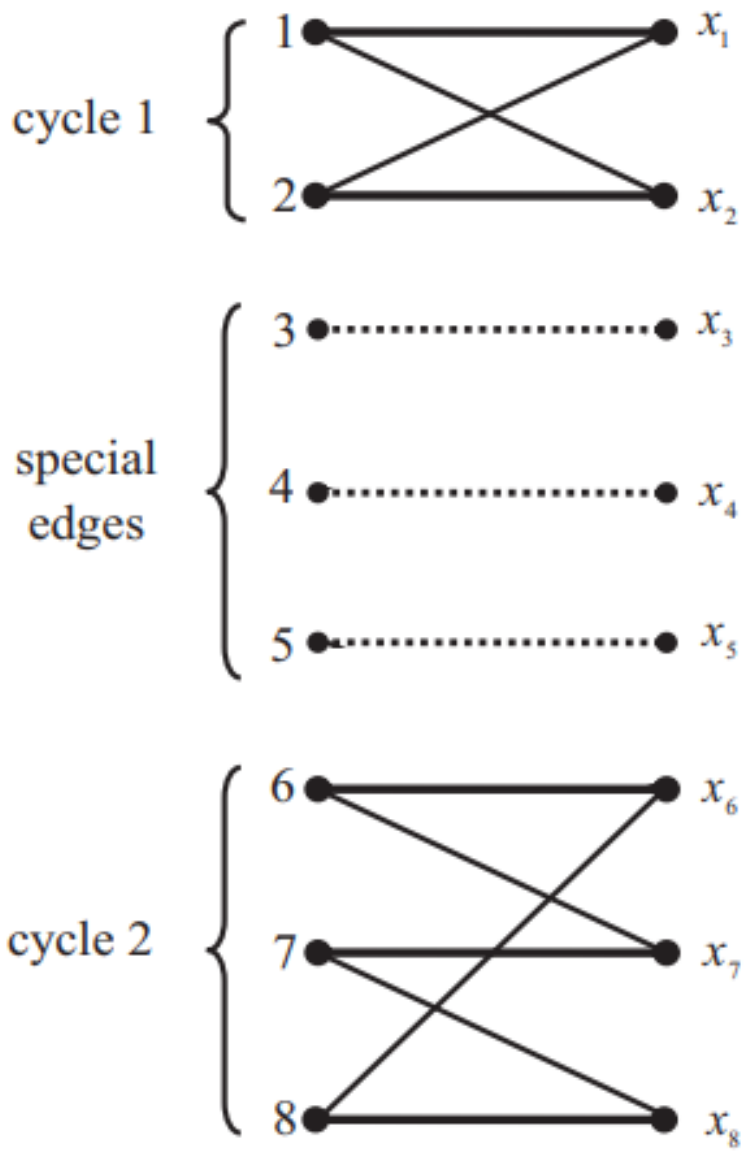
Types of Edges

- An edge $\{i, x\} \in \bar{U}$ is called *special* if $\{i, x\}$ belongs to all perfect matchings in the graph $\bar{G}$.
- All edges, except for the special edges and those adjacent to them, are slit into cycles.

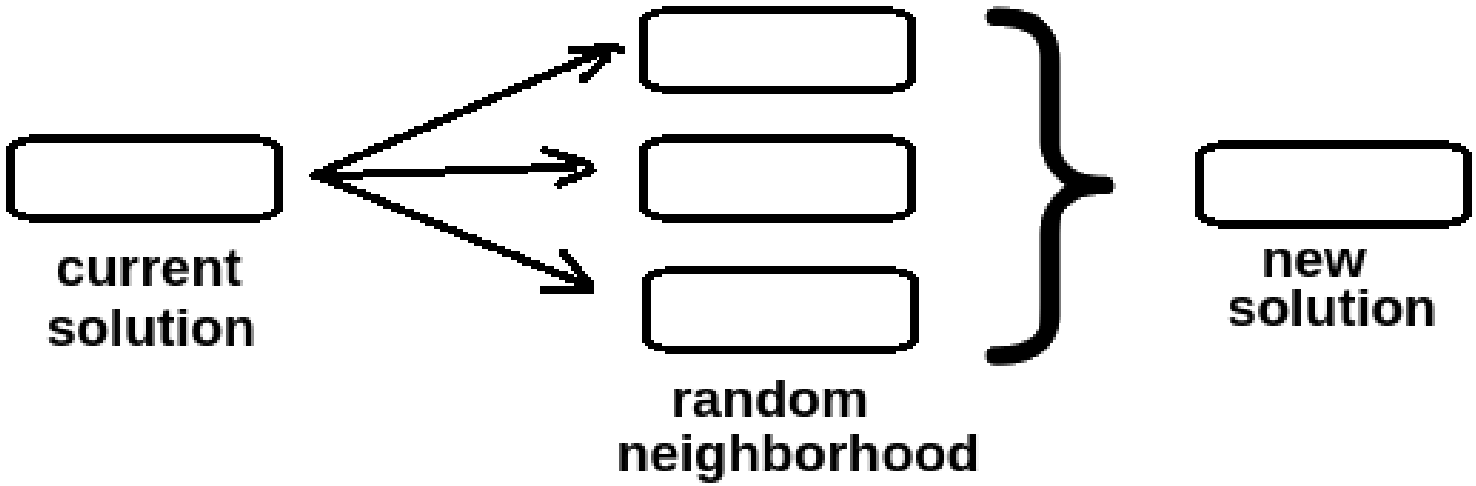
Solving Approach

Step 1. Build the bipartite graph $\bar{G}$, identify the set of special edges and cycles and find all maximal matchings in cycles.
Step 2. Enumerate all perfect matchings $W \in \mathcal{W}$ of $\bar{G}$ by combining the maximal matchings of cycles and joining them with special edges.
Step 3. Assign the corresponding solution $\pi \in \Pi$ to each $W \in \mathcal{W}$ and compute $\gamma(\pi)$.
Step 4. Output the result $\pi^* \in \Pi$, such that $\gamma(\pi^*) = \min\limits_{\pi \in \Pi} \gamma(\pi)$.

Example (ORP Solving)



Parallel (1+λ)-EA



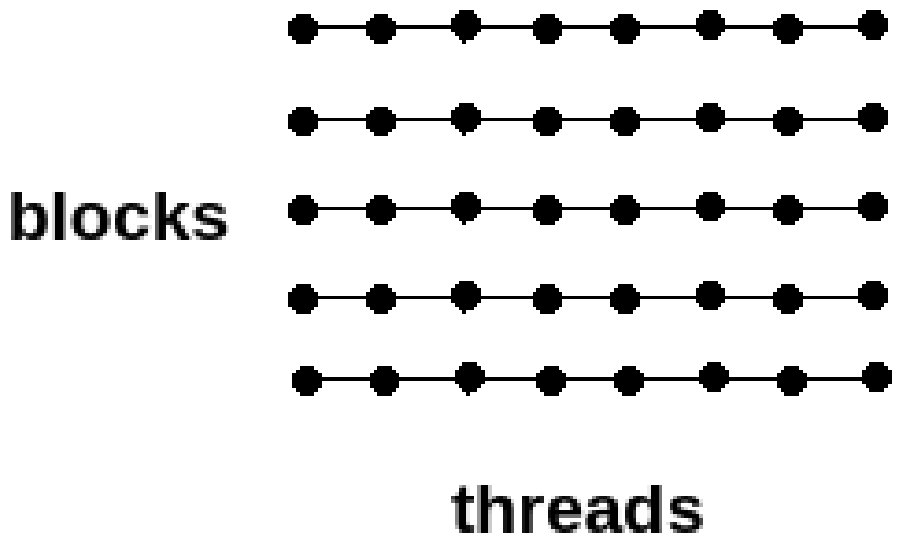
1. Build initial solution π.
2. While stopping criterion is not satisfied:
   2.1. Build λ offspring $\{\sigma_1, \ldots, \sigma_\lambda\}$ applying mutation to π.
   2.2. Let σ be the best offspring among $\{\sigma_1, \ldots, \sigma_\lambda\}$.
   2.3. With probability p replace π by σ and
   with probability $1 - p$ replace π by the best of π and σ.

GPU parallelization

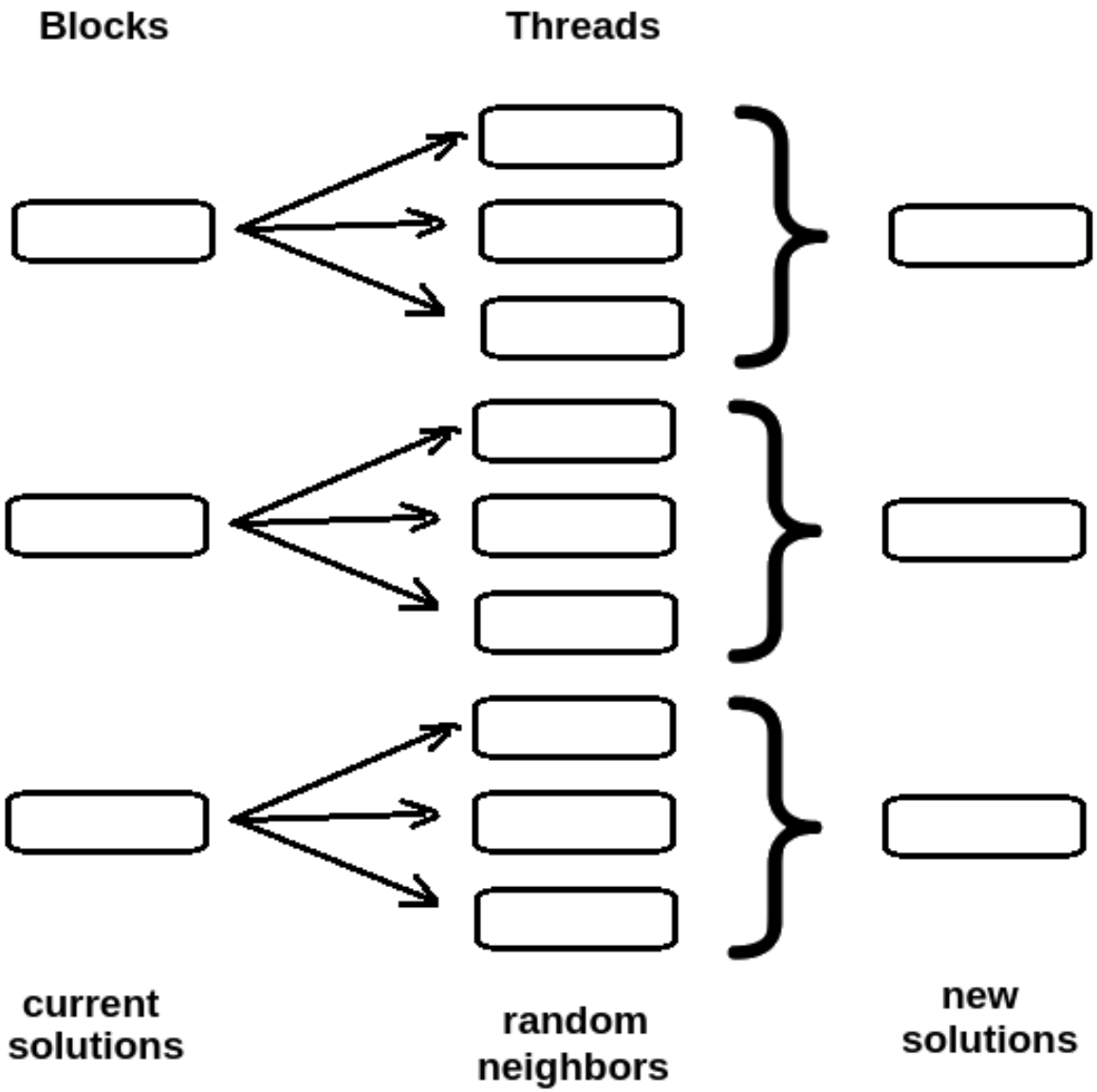
Execute some code on GPU:

```
solve <<< m, n >>> ( ... );
```

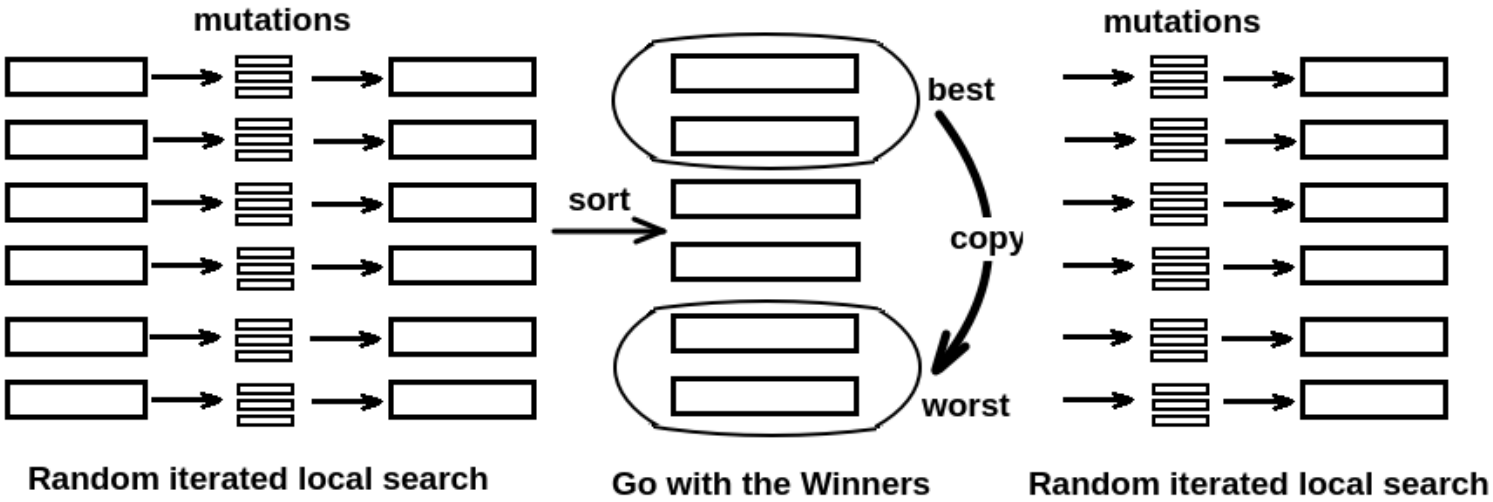
which means m blocks, each one contains n threads.



Many parallel processes of (1+λ)-EA



Hybrid ILS-GwW algorithm



Hybrid ILS-GwW algorithm

1. Build initial solutions $\pi_1, \ldots, \pi_N$.
2. While stopping criterion is not satisfied
   2.1. Run N parallel $(1+\lambda)$-EA processes starting from $\pi_1, \ldots, \pi_N$ with the limit on the number of iterations.
   2.2. Among the current $\pi_1, \ldots, \pi_N$ choose the R best solutions $(\overline{\pi}_1, ..., \overline{\pi}_R)$ and the R worst solutions $(\underline{\pi}_1, ..., \underline{\pi}_R)$.
   2.3. Replace $(\underline{\pi}_1, ..., \underline{\pi}_R)$ by the copies of $(\overline{\pi}_1, ..., \overline{\pi}_R)$.
   2.4. If the shaking condition holds, apply shaking procedure.

Convergence of ILS-GwW

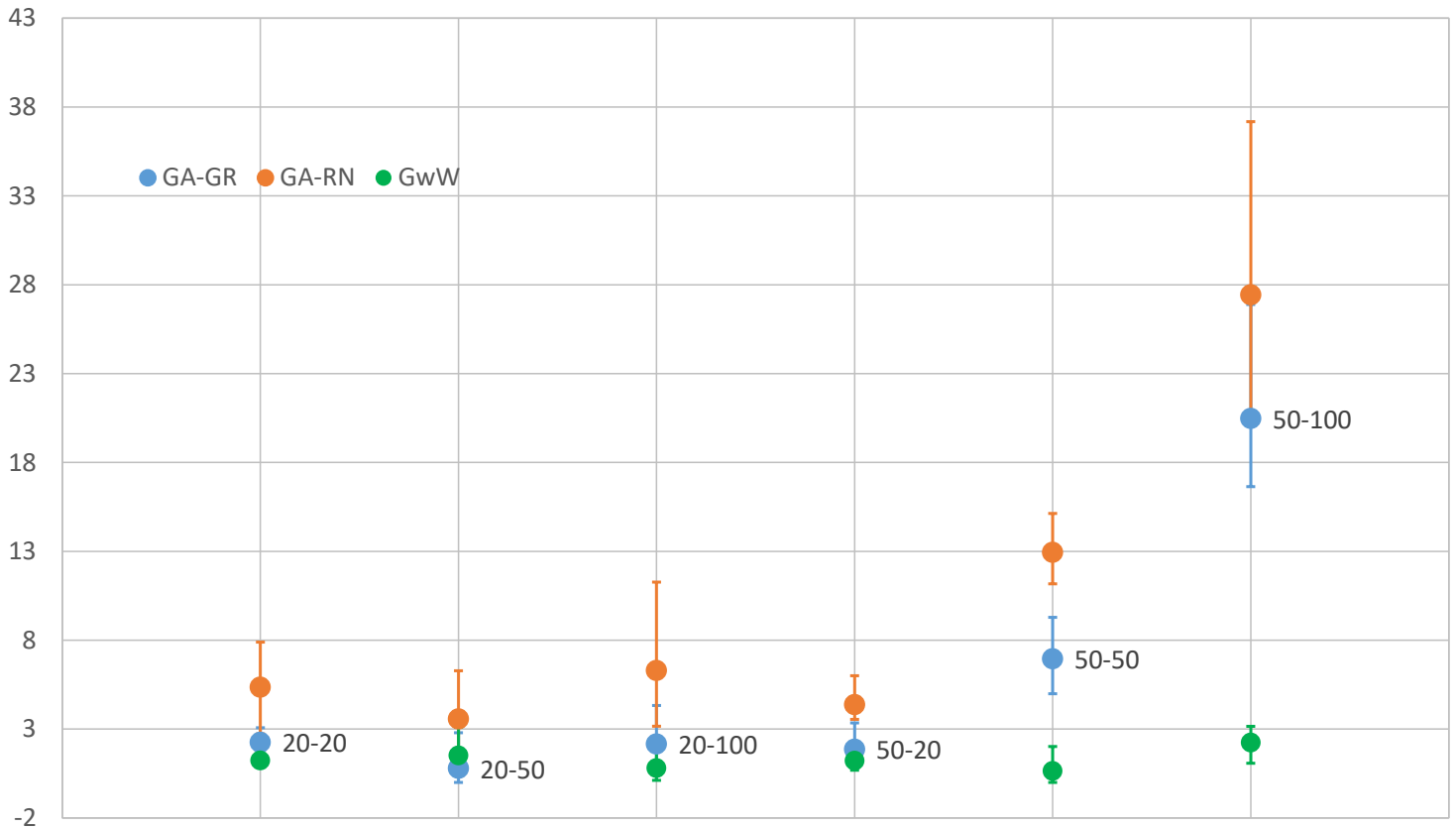
ILS-GwW algorithm almost surely converges to an optimum as the number of iterations tends to infinity.

Results for single machine problem

Series	GA			ILS-GwW		
n-m	Time	GA-GR	GA-RN	Time	CPU	GPU
20-20	1200	2.2530	5.355	800	6.0774	**1.2315**
20-50	1800	**0.7844**	3.4564	1200	7.5514	1.401
20-100	3600	2.1624	6.2971	2400	25.8	**0.8039**
50-20	1800	1.8652	4.3787	1200	7.6236	**1.2252**
50-50	5400	6.9498	12.9446	3600	25.4435	**0.6407**
50-100	7200	20.48	27.4265	4800	88.8814	**2.2458**

The results of ILS-GwW statistically significantly differ from the others with respect to all series.

Results for single machine problem



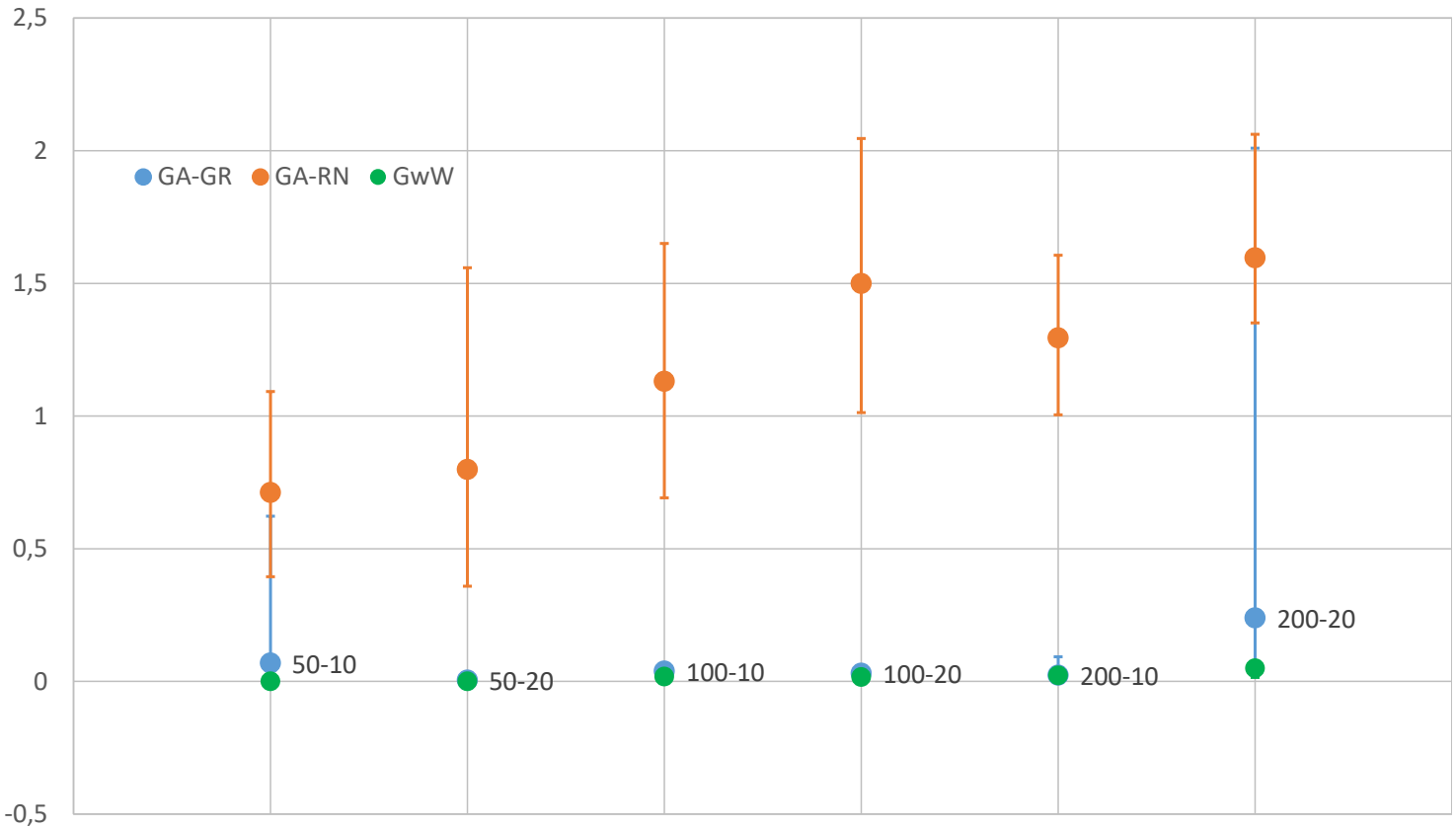
Results for multiple machine problem

Series	LDR-AS [1] and GAs				ILS-GwW		
n-m	Time	LDR-AS	GA-GR	GA-RN	Time	CPU	GPU
50-10	9	0.002	0.0317	0.7117	6	0.1253	**0.0002***
50-20	17	0.005	0.0049	0.7989	12	0.189	**0.0006***
100-10	46	0.0385	0.0385	1.1307	30	0.3007	**0.0179***
100-20	91	0.0307	0.0307	1.4994	60	0.456	**0.016***
200-10	162	0.0232	0.0232	1.2774	108	0.4374	**0.0223**
200-20	324	0.0688	0.2393	1.5967	216	0.723	**0.0491***

*The results of the algorithm statistically significantly differ from the others.

[1] Shi et al A learning-based two-stage optimization method for customer order scheduling. Comput. Oper. Res. (2021)

Results for multiple machine problem

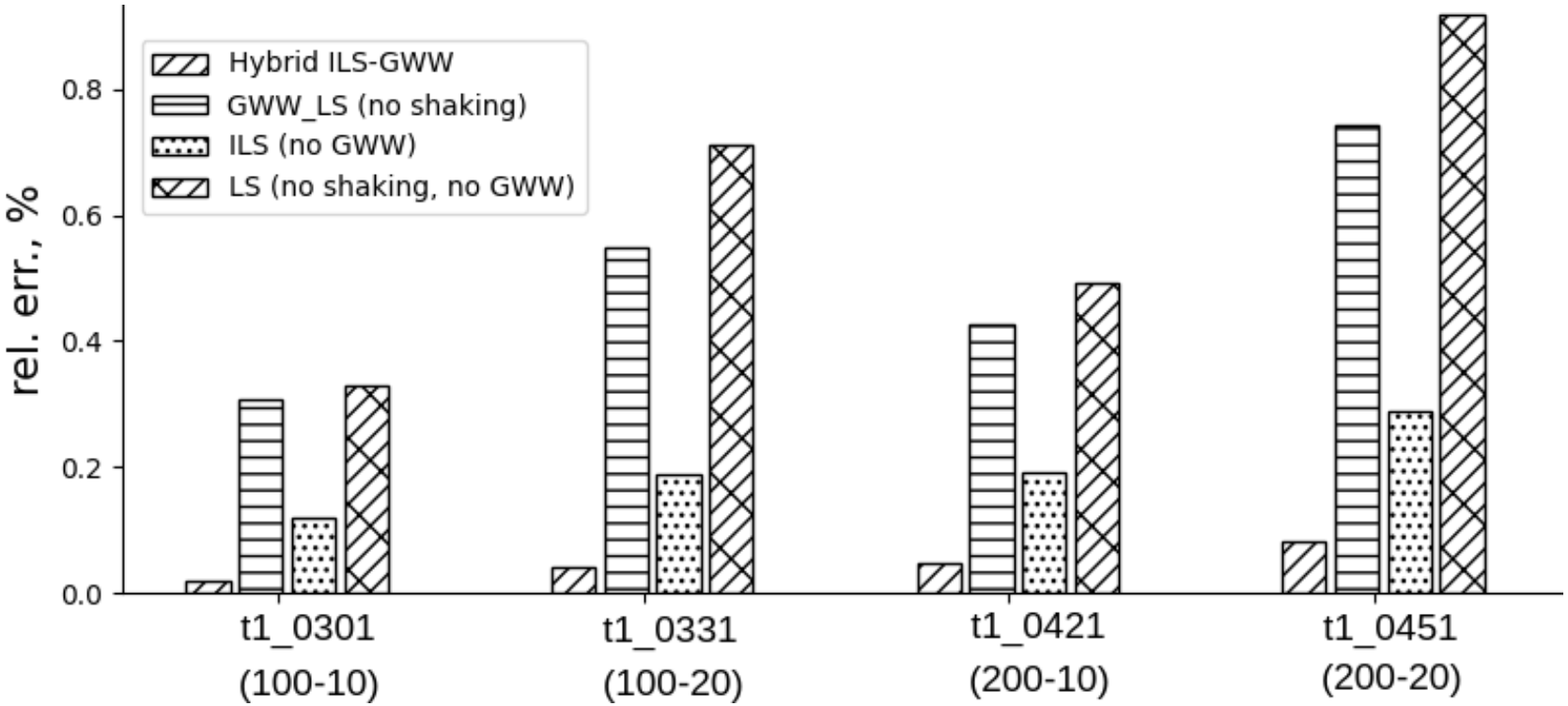


Results for multiple machine problem

New best solutions

n-m	Number of improvements
50-10	7 of 30
50-20	11 of 30
100-10	30 of 30
100-20	30 of 30
200-10	30 of 30
200-20	30 of 30

Hybrid ILS-GwW algorithm



Conclusion

Versions of customer order scheduling are considered.

We propose new evolutionary algorithms for searching near optimal solutions: the Genetic Algorithm with the Optimized Crossover and the parallel hybrid Local Search combined with the "Go with the Winners".

The proposed approaches are quite generic, they exploit little knowledge about the features of the problem and perform similarly or better than the previously known heuristics.

The parallel hybrid heuristic is essentially based on a high-performance computing and provides the best results if implemented on a graphics processor.

Thank you for your attention!