SILESIAN UNIVERSITY OF TECHNOLOGY PUBLISHING HOUSE

SCIENTIFIC PAPERS OF SILESIAN UNIVERSITY OF TECHNOLOGY 2023
ORGANIZATION AND MANAGEMENT SERIES NO. 168

TESTING ALGORITHMS FOR QUICK RESCHEDULING FLOW SHOP PROBLEMS WITH FLEXSIM BASED SIMULATION AND R ENGINE

Piotr JANKE

Silesian University of Technology, Faculty of Organization and Management; piotr.janke@polsl.pl,
ORCID: 0000-0001-8065-9013

Purpose: The aim of this paper is to present a combination of advanced algorithms for finding optimal solutions together with their tests for a permutation flow-shop problem with the possibilities offered by a simulation environment. Four time-constrained algorithms are tested and compared for a specific problem.
Design/methodology/approach: Four time-constrained algorithms are tested and compared for a specific problem. The results of the work realisation of the algorithms are transferred to a simulation environment. The entire solution proposed in the work is composed as a parallel environment to the real implementation of the production process.
Findings: The genetic algorithm generated the best solution in the same specified short time. By implementing the adopted approach, the correct cooperation of the FlexSim simulation environment with the R language engine was obtained.
Research limitations/implications: The genetic algorithm generated the best solution in the same specified short time. By implementing the approach, a correct interaction between the FlexSim simulation environment and the R language engine was achieved.
Practical implications: The solution proposed in this paper can be used as an environment to test solutions proposed in production. Simulation methods in the areas of logistics and production have for years attracted the interest of the scientific community and the wider industry. Combining the achievements of science in solving computationally complex problems with increasingly sophisticated algorithms, including artificial intelligence algorithms, with simulation methods that allow a detailed overview of the consequences of changes made seems promising.
Originality/value: The original concept of cooperation between the R environment and the FlexSim simulation software for a specific problem was presented.

Keywords: flow-shop problem, genetic algorithm, simulation.

Category of the paper: Research paper.

1. Introduction

The scheduling of production orders is one of the most important tasks of companies producing finished goods. One of the basic combinatorial tasks in the scheduling of production orders is the so-called flow-shop problem. In this problem, the sequence of operations on the machines is fixed. All orders pass through all machines without repeating operations. The aim of such scheduling is to work out the sequence of orders execution on the machines in such a way that the execution time of all orders (makespan) is as short as possible. In this way, we are dealing with an optimisation task that can be presented as a permutation problem. The literature on the subject also describes a situation where the flow-shop problem is treated non- permutationally. (Rossit, Tohmé, Frutos, 2018). Depending on the number of jobs (n) and the number of machines (m), the degree of difficulty varies, resulting from the size of the number of permutations to be searched. In the literature, tasks of this type are designated as tasks of NP-hard. In the case of different sequences of tasks on the machines for individual jobs, we are dealing with the so-called jobshop problem. The literature also describes an open-shop problem where no sequence has to be followed. Over the years, quite a few scientific publications have been written on this topic. These problems are solved more or less successfully by well-known metaheuristics such as simulated annealing, tabu search or genetic algorithms (Rolf, Reggelin, Nahhas, Lang, Müller, 2020). Various approaches are used. In works (Ramesh, Kamalakannan, Karthik, Pavin, 2020) simulated annealing was used to optimise the total schedule execution time in a flow-shop problem. The use of genetic algorithms for production scheduling flow-shop problems can also be found in the works of Polish researchers such as: Ławrynowicz (2011), Pawlak (1999), Knosala, Wal (2001). Changes in the limiting conditions or modifications within the operation of the genetic algorithm are very common in scientific papers. In the papers (Andrade, Silva, Pessoa, 2019) a modification of the genetic algorithm to avoid local optima more efficiently is proposed. Further modifications to improve selected algorithms are presented in papers such as: (Jankauskas, Papageorgiou, Farid (2019), Huynh & Chien (2018), Touat, Bouzidi-Hassini, Benbouzid-Sitayeb, & Benhamou (2017), Guido & Conforti (2017). These are just a selection from of the many publications on this subject. The literature on the subject is also rich in publications comparing the performance of selected algorithms for a specific class of problems. In parallel, methods of optimising production processes using event-based simulation tools are becoming equally popular. A comparison of both approaches in solving a flow-shop problem is presented in papers (Kaczmar, Bányai, 2022). This paper proposes a combination of both techniques.

The primary objective of the paper is to compare the performance of selected algorithms for the optimisation of a finite size task in finite (specified) time for the same objective function. In addition, the paper proposes a concept for the interaction of the R engine with an event-driven simulation environment for the analysis of the obtained solutions.

2. Methodology

The problem of scheduling tasks (production orders) presented as a scheduling optimisation task for minimising the execution time of all orders (MakeSpan Cmax -> min) can be solved using different methodologies. As mentioned in the introduction, metaheuristics are popular and frequently used. This paper focuses on the permutation flow-shop problem. These algorithms often balance between exploiting a selected space within a defined neighbourhood and exploring to 'refresh' these areas. The test environment in the study is the R language version 4.2.1 on a virtual machine running in a HyperV environment with 32 virtual cores and 20 GB of RAM allocated. The hypervisor is a dual-processor Xeon E5-2620 with RAID 1 on SSDs. The selected algorithms are from packages made available in the CRAN repository. As most of the algorithms used in the study do not run natively on permutations, it was decided to use their own implementations.

None of the algorithms tested had parallelization computation mechanisms running.

As the simulation environment does not perform any complex calculations it can be run on any hardware configuration that meets the minimum requirements of the simulation software.

The proposed approach is as follows:

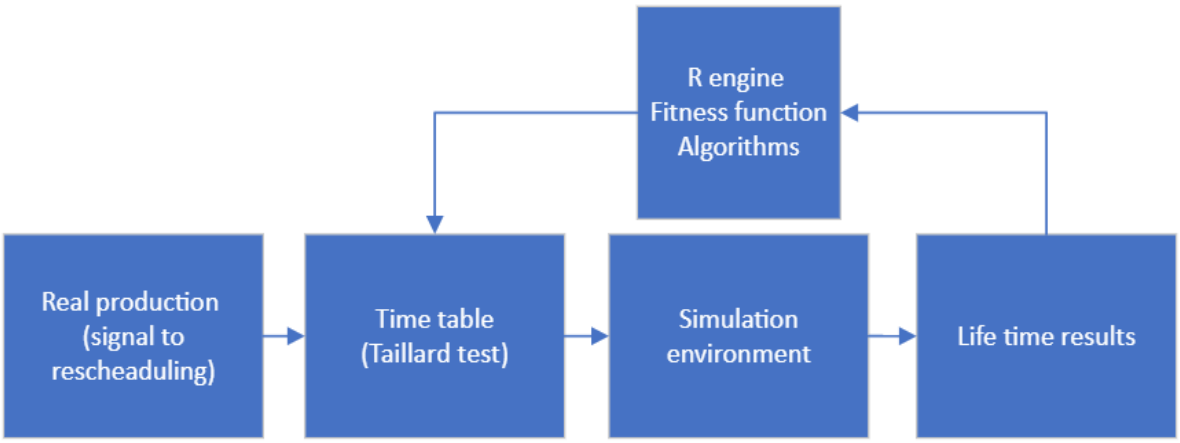

Figure 1. The proposed approach to implementing the environment.

Source: Own study.

The need for changes is implemented through a signal from real production updating the time tables. Changes in the production schedule change the results of the simulation execution. These results, via a trigger, activate calculations on the R side of the language to eventually return a new optimised time table at finite output.

2.1. Algorithms

The comparative study first used the well-known methodology of creating new permutations of a set allowing all of them to be found without the possibility of repetition.

For this purpose, the chosen 'perm' function of the Combinat package was selected (Carey, 2015) modifying it so that in each successive iteration it is possible to generate another permutation to check the objective function. Originally, the function returned all permutations of the set.

The next algorithm created random permutations using the 'sample' function analogously to the previous one by checking the value of the objective function.

The third algorithm was simulated annealing – own implementation.

The last algorithm tested was the genetic algorithm, which is an original implementation for the R language from the GA package (Scrucca, 2013). This implementation was the only one in the study that could work natively on permutations.

2.2. Data set

In the comparative study of algorithms, it was decided to choose a problem for the number of jobs n = 20 and the number of machines m = 10 and set number 8 from Taillard's test field data set also called benchmarks for scheduling problems (Taillard, 1993).

Table 1.
Taillard test - set number 8 for 20 jobs

	X	JOBS	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
1	1	JOB1	9	91	96	73	37	28	32	27	4	83
2	2	JOB2	19	54	78	88	94	58	18	63	67	4
20	20	JOB20	76	4	24	74	19	46	27	82	26	63

Source: Own study.

The changeover times in this set are neglected. The space of all solutions, i.e. the number of all permutations for the jobs in this problem, is of size 20 factorial. The flow-shop problem for the Taillard test under study can be represented by a simulation model of the FlexSim software (Fig. 2).

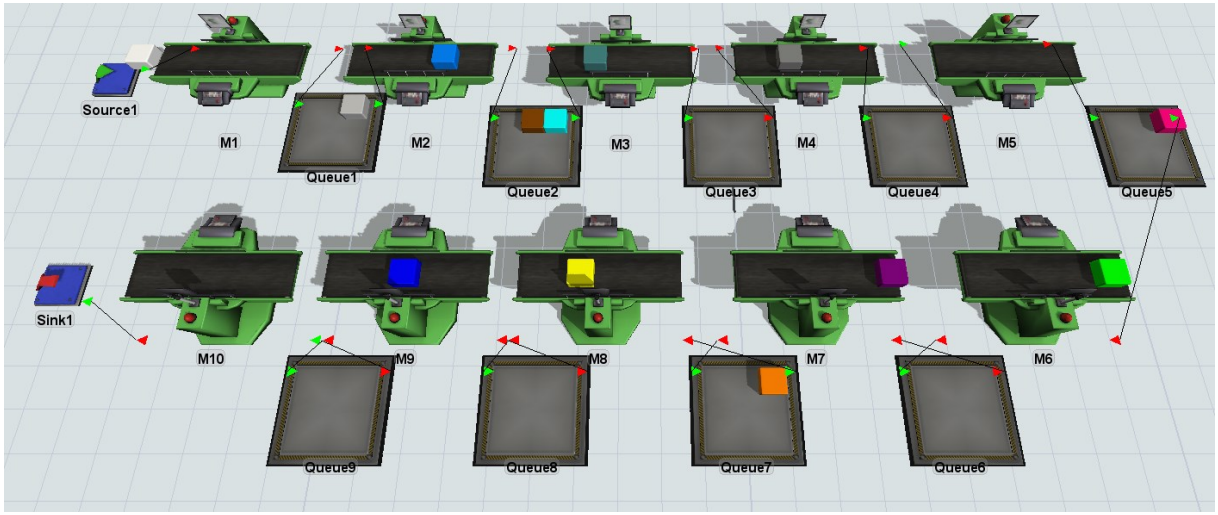

Figure 2. FlexSim simulation model for a 20x10 flow-shop problem.

Source: Own study.

The buffer fields are represented as queues between the machines. The processing times of the workpieces on the individual machines "Processing Time" depend directly on the job number. This relationship is shown as a reference to the corresponding cell in the global table of the Taillard test imported into it:

Table("Taillard20_10x8 ")[**JobNumber**][MachineNumber]

The material processing times on all machines for the default order sequence of 1 to 20 are respectively:

Table 2.
Completion times on all machines of each job for the default sequence

Results

JobNumber	SimulationTime
1.00	480.00
2.00	666.00
3.00	789.00
4.00	863.00
5.00	909.00
6.00	957.00
7.00	1064.00
8.00	1101.00
9.00	1236.00
10.00	1247.00
11.00	1305.00
12.00	1336.00
13.00	1401.00
14.00	1483.00
15.00	1571.00
16.00	1663.00
17.00	1716.00
18.00	1886.00
19.00	1945.00
20.00	2044.00

Source: Own study.

For the default order (permutation) in this task, the total completion time for all jobs on all machines equals the completion of the last job and is 2044 units.

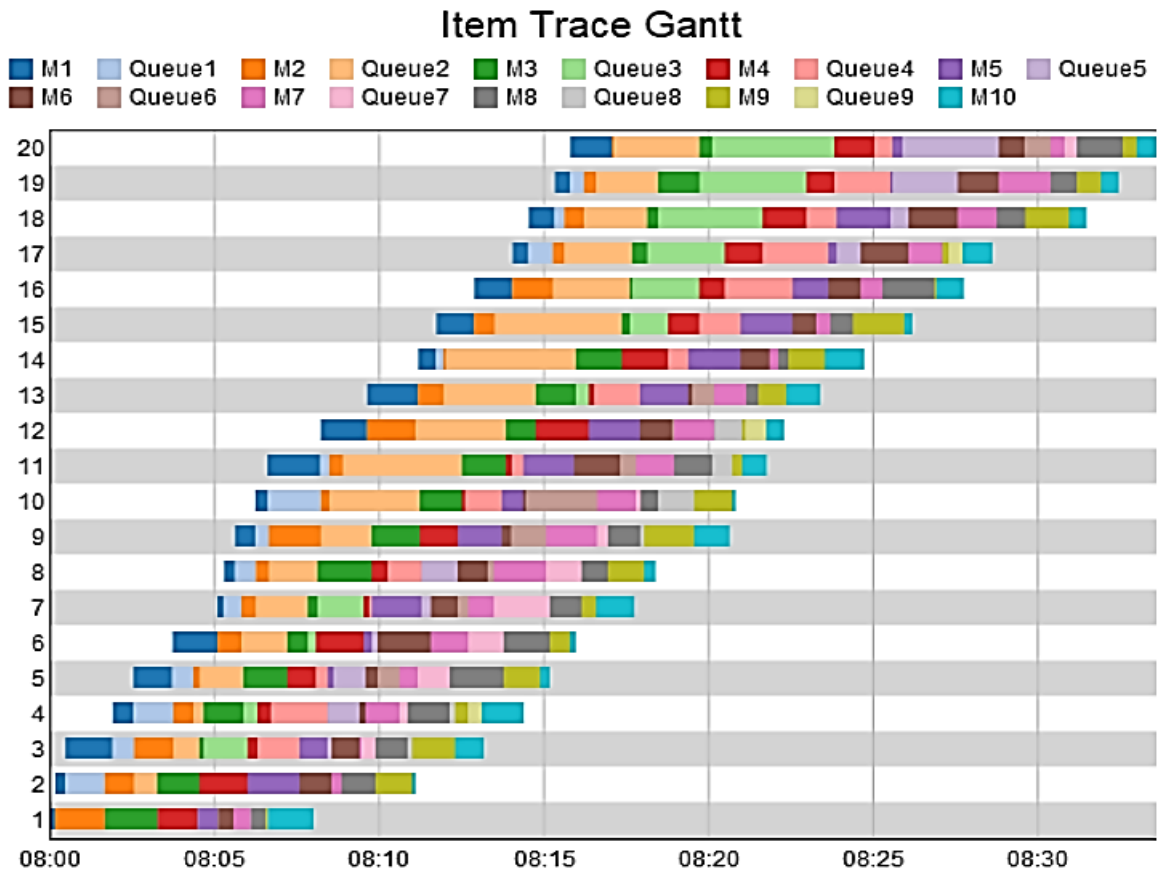

Figure 3. Job-machine and queue Gantt chart for the default sequence.

Source: Own study.

As previously mentioned, the queues in the simulation model correspond to the buffer fields that allow the machine to be released after the job is done.

2.3. Fitness function

The total and maximum execution time of the whole schedule (makespan) is calculated using the following parameters (Pang, Xue, Tseng, Lim, Liu, 2020):

- $t(\sigma j, k)$ = processing time for job j on machine k (j = 1, 2, 3, …, n), (k = 1, 2, 3, …, m).
- n = total number of jobs to be processed.
- m = total number of machines in the process.
- $\sigma = \{\sigma 1, \sigma 2, \ldots, \sigma n\}$ = permutation job set.

Makespan can be formulated as follows:

- $C(\sigma 1, 1) = t(\sigma 1, 1)$,
- $C(\sigma j, 1) = C(\sigma j - 1, 1) + t(\sigma j, 1)$ where j = 2, 3, …, n,
- $C(\sigma 1, k) = C(\sigma 1, k - 1) + t(\sigma 1, k)$ where k = 2, 3, …, m,
- Cmax = $C(\sigma j\ , k) = \max(C(\sigma j - 1, k), C(\sigma j, k - 1)) + t(\sigma j, k)$.

3. Results

Due to the large solution space in the proposed approach, calculations can be performed on the R engine side and then imported as a time table natively into FlexSim software.

The first will present the values of the matching function for an algorithm that searches all permutations (first from the right) for a fixed order. The second algorithm is random search. The third Simulated annealing and the fourth a genetic algorithm. The results are presented on fig. 4 for an assumed 100 seconds of operation in the same environment and with the same parameter seed fixed.

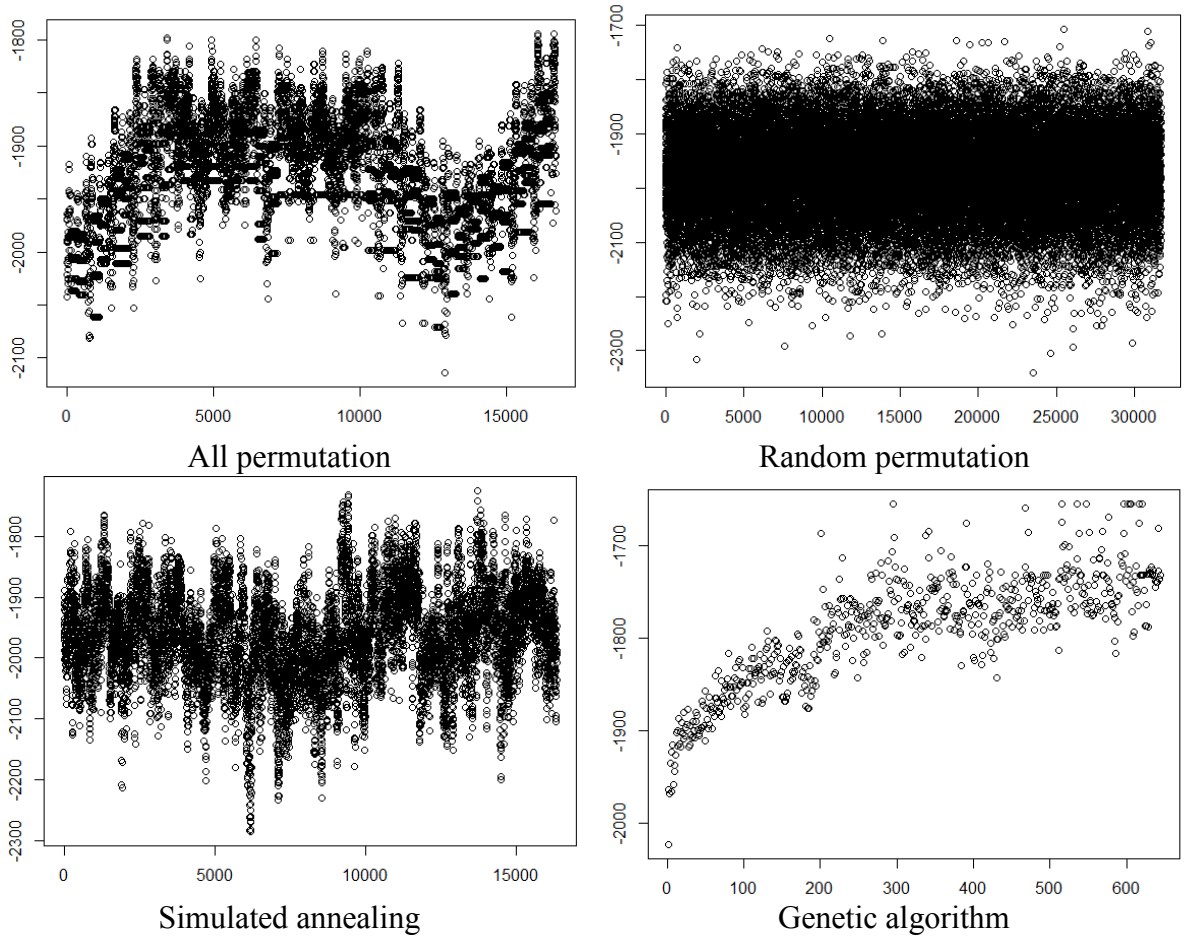

Figure 4. All fitness function values for all iterations of the running algorithms.

Source: Own study.

On the horizontal axis is the number of iterations of the running algorithm during the fixed 100 seconds of operation. On the vertical axis, the values of the objective function. The graphs were generated using the "plot" function of the R language.

As can be seen, the space available per 100 seconds for all running algorithms varies. Due to the complexity and thus the number of necessary calculations performed in iterations, the algorithms also differ in the number of iterations performed in the same time. The leader in this respect is the random search algorithm whose calculations in a single iteration only require

drawing permutations and calculating the matching function. An algorithm that calculates permutations according to a fixed order and allows all available permutations to be generated without repetition takes on average twice as long as random search in a single iteration. In the case of the simulated annealing algorithm, the amount of computation per iteration is greater. In each iteration, the algorithm searches a defined neighbourhood, compares the results and determines the temperature to decide on exploration or exploitation on this basis. The genetic algorithm is the most computationally advanced in this comparison.

The next graphic (Fig. 5) shows a comparison of the results of the algorithms keeping the best solutions (Cmax ->min) in relation to all preceding ones. In this way, the convergence of the algorithms in iterations can be compared and evaluated.

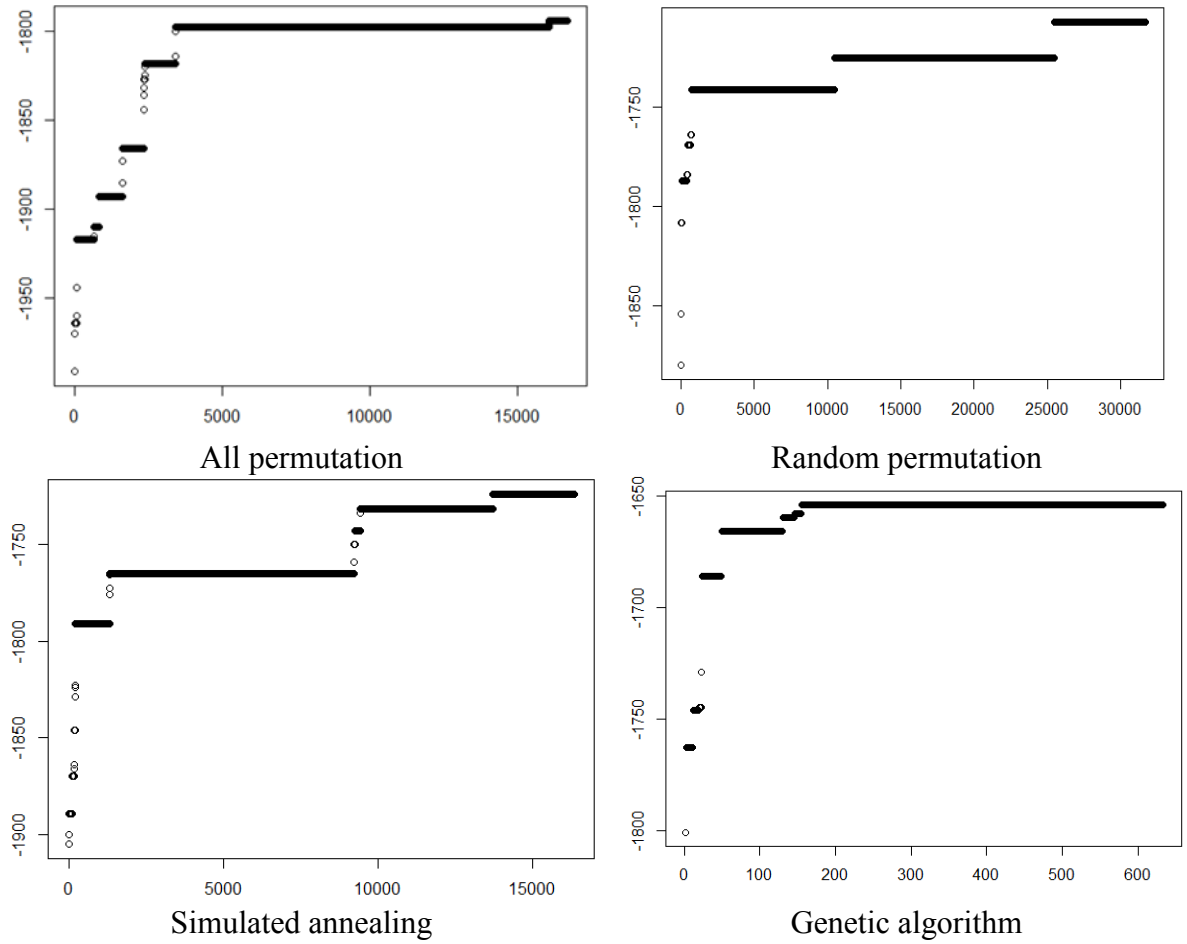

Figure 5. Best fitness function values for all iterations of the running algorithm.

Source: Own study.

The genetic algorithm, among those being compared, is the fastest converging to the best solution. This happens both in time and, of course, in the number of iterations.

The next table shows the results of the obtained parameters over the set time:

Table 1.
All finall results with 100 second running algorithms

Algorithm	Iterations	Fitness Value
All permutation	16800	1794
Random permutation	31663	1707
Simulated annealing	16359	1724
Genetic algorithm	633	1654

Source: Own study.

Each of the solutions found is better than the default solution. The genetic algorithm was able to find the best solution among the available solutions for the algorithms tested in 100 seconds. Best chromosome selected for the values of the matching function: job8, job19, job7, job18, job9, job1, job5, job15, job20, job13, job6, job11, job14, job4, job12, job3, job10, job17, job2, job16.

The performance parameters of the genetic algorithm are as follows:

GA settings:

- Type = permutation.
- Population size = 100.
- Number of generations = 5000.
- Elitism = 50.
- Crossover probability = 0.8.
- Mutation probability = 0.1.

The results of a simulation of one of the best solutions found by the genetic algorithm are shown below.

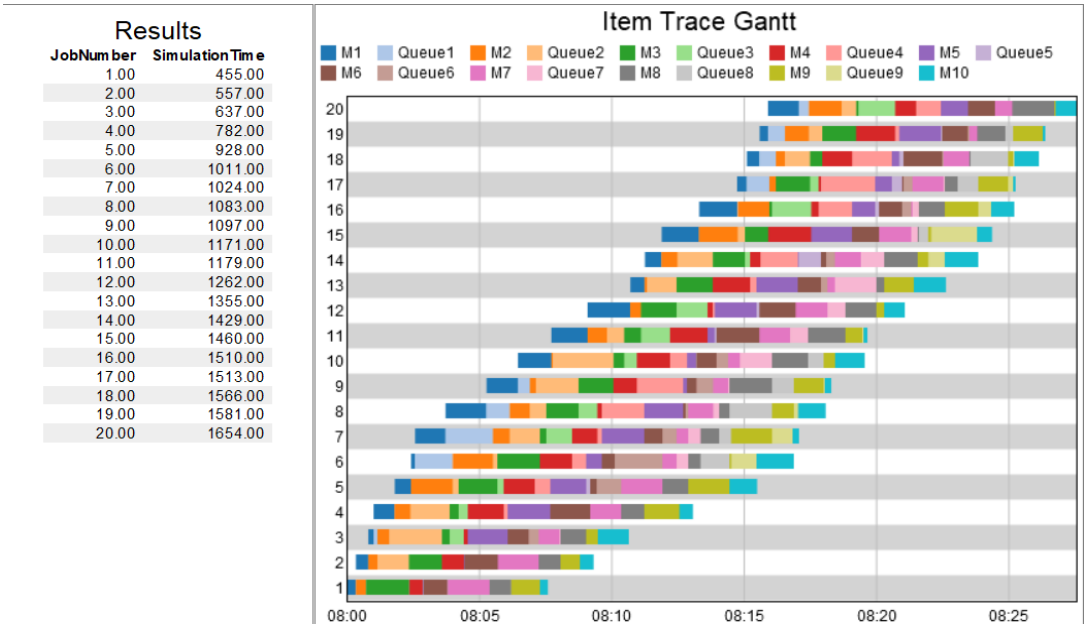

Figure 6. Completion times on all machines of each job and Gantt chart in job-machine and queue layout (time unit seconds).

Source: Own study.

The sequence selected by the genetic algorithm was imported as a time table. The results of the simulation coincide with the values of the matching function obtained from the genetic algorithm. Completion times on all machines of each order are shown in a Gantt chart in the job-machine layout and the queues associated with the job waiting before the machine. The next graphic (Fig. 7) compares the Gantt charts with the orders on the machines in the second layout in both environments. The top part comes from the FlexSim simulation software and the bottom part is the result of an algorithm on the R language side.

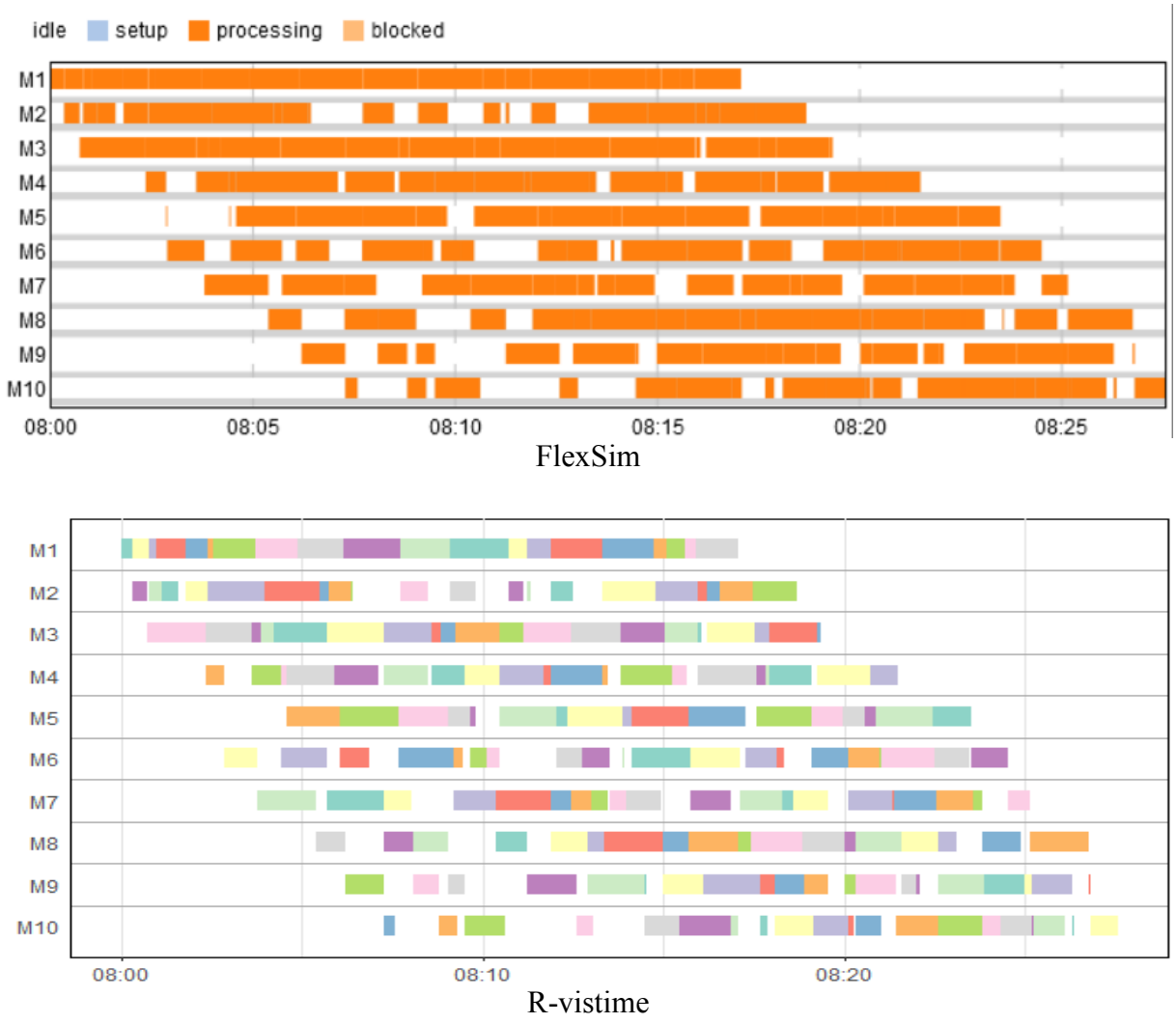

Figure 7. Comparison of Gantt charts made in FlexSim and R software (time unit seconds).

Source: Own study.

In the case of the FlexSim simulation, the "Gantt State" template was used to plot the chart and, in the case of the R language, the "vistime" package. Both the execution times on the machines and the gaps between jobs are arranged in the Gantt chart in the same way, representing the same schedule.

4. Discussion

In reality, the scheduling problem of an actual production is very rarely presented as a single type of task. Nevertheless, in an actual production scheduling it is possible to find a place where some order influences the final completion time of all jobs. This may be, as in this paper, the sequence of jobs with different execution times on the machines, the sequence of tooling for retooling a particular machine or permutations can be made with job-shop problems with the machines themselves. The paper shows that it is possible to represent a flow-shop problem in the form of an event-driven simulation with data from external sources. The paper uses a Taillard test of a certain size, which can be replaced by any data table. The results of the implementation of the algorithm on the R engine side, i.e. the values of the fit function (objective function) obtained from the best achieved permutation of jobs, are identical to the values obtained from the simulation. The simulation experiment does not evaluate the solution but only represents the execution process of the jobs on the basis of a time table. The paper does not compare the performance of algorithms written in R language with those that would be performed in simulation software. Instead, a combination of one and the other is presented. This solution undoubtedly has the advantages of making the calculations independent of the simulation itself, which can allow implementation in an environment unconstrained by the requirements of simulation software. The results from one source can be fed back in parallel to the real production as well as to the digital twin of that production in the simulation environment. The advantage of using simulation is undoubtedly that, in this case, it is possible to trace, step by step, the events and consequences of the changes introduced into production.

4. Conclusions

The four algorithms were tested for a limited time of 100 seconds in the R engine with identical hardware configuration and the same seed parameter. The following algorithms were tested sequentially: random permutations, all according to a fixed order, a simulated annealing algorithm and a genetic algorithm. The test field was the chosen Taillard test for the flow problem with permutations. The results of the algorithms and the proposed concept served as input for a simulation experiment in FlexSim software. A simulation model for the flow problem conforming to the 20x10 test was developed and the obtained results of the algorithms' fit function were verified with the result of the simulation.

References

1. Andrade, C.E., Silva, T., Pessoa, L.S. (2019). Minimizing flowtime in a flowshop scheduling problem with a biased random-key genetic algorithm. E*xpert Systems with Applications, 128,* 67-80. https://doi.org/https://doi.org/10.1016/j.eswa.2019.03.007.
2. Carey, M.V. (2015). Package 'combinat'.
3. Guido, R., Conforti, D. (2017). A hybrid genetic approach for solving an integrated multi-objective operating room planning and scheduling problem. Co*mputers & Operations Research, 87,* 270-282. https://doi.org/https://doi.org/10.1016/j.cor.2016.11.009.
4. Huynh, N.-T., Chien, C.-F. (2018). A hybrid multi-subpopulation genetic algorithm for textile batch dyeing scheduling and an empirical study. Com*puters & Industrial Engineering, 125,* 615-627. https://doi.org/https://doi.org/10.1016/j.cie.2018.01.005.
5. Jankauskas, K., Papageorgiou, L.G., Farid, S.S. (2019). Fast genetic algorithm approaches to solving discrete-time mixed integer linear programming problems of capacity planning and scheduling of biopharmaceutical manufacture. Co*mputers & Chemical Engineering, 121,* 212-223. https://doi.org/https://doi.org/10.1016/j.compchemeng.2018.09.019.
6. Kaczmar, I., Bányai, T. (2022). Optimisation of flow shop scheduling problem: simulation system vs. evolutive solver. Adva*nced Logistic Systems - Theory and Practice, 16(1),* 31-40. https://doi.org/10.32971/als.2022.003.
7. Knosala, R., Wal, T. (2001). Production scheduling problem using genetic algorithm. Journ*al of Materials Processing Technology, 109(1-2),* 90-95. https://doi.org/10.1016/S0924-0136(00)00780-9.
8. Ławrynowicz, A. (2011). Genetic Algorithms for Solving Scheduling Problems in Manufacturing Systems. *Foundations of Management, 3(2),* 7-26. https://doi.org/10.2478/v10238-012-0039-2.
9. Pang, X., Xue, H., Tseng, M., Lim, M.K., Liu, K. (2020). Ap*plied Sciences Improved Fireworks Algorithm for Permutation*.
10. Pawlak, M. (1999). *Algorytmy ewolucyjne jako narzędzie harmonogramowania produkcji.* Warszawa: PWN.
11. Ramesh, C., Kamalakannan, R., Karthik, R., Pavin, C. (2020). A lot streaming based flow shop scheduling problem using simulated annealing algorithm. *Materials Today: Proceedings.* https://doi.org/https://doi.org/10.1016/j.matpr.2020.05.108.
12. Rolf, B., Reggelin, T., Nahhas, A., Lang, S., Müller, M. (2020). Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem. *Procedia Manufacturing, 42,* 442-449. Elsevier B.V. https://doi.org/10.1016/j.promfg.2020.02.051.
13. Rossit, D.A., Tohmé, F., Frutos, M. (2018). The Non-Permutation Flow-Shop scheduling problem: A literature review. *Omega, 77,* 143-153. https://doi.org/10.1016/J.OMEGA.2017.05.010.

14. Scrucca, L. (2013). GA: A Package for Genetic Algorithms in R. *Journal of Statistical Software, 53(4 SE-Articles),* 1-37. https://doi.org/10.18637/jss.v053.i04.
15. Taillard, E. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research, 64(2),* 278-285. https://doi.org/10.1016/0377-2217(93)90182-M.
16. Touat, M., Bouzidi-Hassini, S., Benbouzid-Sitayeb, F., Benhamou, B. (2017). A hybridization of genetic algorithms and fuzzy logic for the single-machine scheduling with flexible maintenance problem under human resource constraints. *Applied Soft Computing, 59,* 556-573. https://doi.org/https://doi.org/10.1016/j.asoc.2017.05.058.