# Multi-factory Job Shop Scheduling With Due Date Objective

Jacob Lohmer[1], Daniel Spengler[1], Rainer Lasch[1]
[1]Chair of Business Management, esp. Logistics, Technische Universität Dresden, Dresden, Germany
*jacob.lohmer@tu-dresden.de*

*Abstract* –**The existing literature on distributed scheduling mainly focuses on the performance measures makespan, total completion time, or costs. Due date related objectives that are gaining in importance in the industry have not been considered to a similar extent. In this contribution, we present a model formulation of the distributed job shop scheduling problem with due date consideration and present adapted greedy heuristics as well as a genetic algorithm to solve large problem instances. Computational experiments are carried out to assess the performance of the model and the algorithms. The heuristics and metaheuristics show promising results in reasonable computation times, with the genetic algorithm outperforming the other heuristics. The results indicate that managers should consider incorporating due date related objectives in the decision-making process of production planning and scheduling in distributed manufacturing.**

*Keywords* – **Multi-factory scheduling, distributed scheduling, distributed job shop, due date objective, greedy heuristic, genetic algorithm**

## I. INTRODUCTION

Manufacturing firms are increasingly distributed in geographical terms and operate multiple factories, as a result of globalization and increased customer demands for fast order delivery [1], [2]. This decentralization of production facilities offers advantages such as lower costs, proximity to customers, and flexibility but also poses new challenges regarding planning and scheduling of production in the factories. Scheduling orders in multi-factory networks involve more steps than the already complex task of scheduling jobs in a single factory [2], [3]. Before sequencing jobs on machines locally, the orders (as jobs) need to be assigned to a factory. The problems vary in complexity, depending on the manufacturing principle or configuration, similar to the single-factory case. This article focuses on the job shop configuration, which is typical for various industries and also present in distributed networks [4], [5]. In a job shop, a set of $n$ jobs have to be manufactured, each with its own routing on the $m$ machines. Single-factory job shop scheduling has been studied extensively by scholars ever since the heuristic by Jackson in 1956 [6]. Distributed scheduling has recently gained the attention of scholars, for recent reviews refer to [2], [7].

We provide a short review of relevant studies for this article that have been published on the multi-factory or distributed job shop scheduling problem (DJSP): Jia et al. [8] developed genetic algorithms (GA) to minimize makespan and costs. Chan et al. examined the DJSP with flexible routings and makespan objective with a GA [9]. Then, [10] focused on a GA to minimize the total completion time of all jobs in the DJSP. In [11], a tabu search and simulated annealing algorithm were hybridized. Two models and three greedy heuristics were proposed for the DJSP with a makespan objective in [12]. Wu et al. [13] examined different chromosome representations in GAs for the DJSP with a makespan objective.

As this short and selective review indicates, previous studies have primarily focused on the performance measures makespan ($C_{max}$) or total completion time ($\sum C_j$). However, the manufacturing industry is evolving quickly and trends like Industry 4.0 led to a situation where for many manufacturing firms, timely deliveries without delays are at least as important as optimal utilization of machine capacities and swift completion of production orders ($C_{max}$) [14]. Earliness and tardiness of orders have different effects on the present and future performance of firms (see Fig. 1). Both objectives should be addressed in order to reach an optimal control of production towards meeting due dates.

To advance in this promising research area, we formulate the problem as a MILP model with a due date objective in this contribution and adapt several heuristics from the literature on classical job shop scheduling for the DJSP Additionally, we propose two adapted greedy heuristics and a genetic algorithm to solve large problem instances efficiently. Experiments are conducted to evaluate the performance of the algorithms using the well-known instances of Taillard [15] and Fisher and Thompson [16].

The remainder of this paper is structured as follows: In Section 2, the mathematical model is presented and described. Section 3 contains the adapted and developed algorithms and heuristics, whereas Section 4 presents numerical experiments and results. Section 5 concludes the paper with an outlook on further research opportunities.

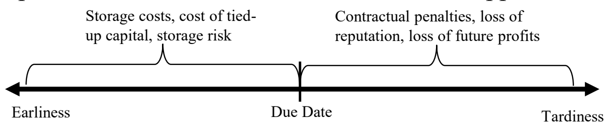
Storage costs, cost of tied-up capital, storage risk
Contractual penalties, loss of reputation, loss of future profits
Earliness
Due Date
Tardiness

Fig. 1. Cost effects (excerpt) of due date violation

## II. PROBLEM DEFINITION AND MATHEMATICAL MODEL

The problem under consideration is modeled as a mixed-integer linear programming model, which adopted the processing constraints (7) and (8) from [12]. Each machine can only process one job at the same time. Although transporting work-in-process orders to another factory in case of disruptions or loss of machine capacity may be possible, it is in most cases not economically feasible and not permitted in the model. The factories are homogeneous and equipped in the same way. Demand is deterministic and static. The following list of notation is used in the model:

| | |
|---|---|
| $J$ | Number of jobs $i, j = \{1, 2, \dots, J\}$ |
| $M$ | Number of machines $m$, $n = \{1, 2, \dots, M\}$ |
| $S$ | Number of factories $s = \{1, 2, \dots, S\}$ |
| $p_{i,m}$ | Processing time of job $i$ on machine $m$ |
| $a_{i,m,n}$ | Binary variable: 1 if for job $i$ machine $m$ is used immediately after machine $n$, 0 otherwise |
| $D_i$ | Due date of job $i$ |
| $L$ | A large positive number |

The decision variables are as follows:

| | |
|---|---|
| $X_{i,j,m}$ | Binary variable: 1 if job $j$ is processed on machine $m$ after job $i$, 0 otherwise |
| $Y_{i,s}$ | Binary variable: 1 if job $i$ is assigned to factory $s$, 0 otherwise |
| $C_{i,m}$ | Continuous variable for the completion time of the operation of job $i$ on machine $m$ |
| $C_{max,i}$ | Continuous variable for the completion time of job $i$ |
| $E_i$ | Earliness of job $i$ |
| $T_i$ | Tardiness of job $i$ |
| V | Total due date deviation of the schedule |

The MILP model is as follows:

$$Minimize\ V = \sum_{i=1}^{J}(E_i + T_i) \quad (1)$$

Subject to:

$$C_{max,i} = \max_m C_{i,m} \qquad \forall i \quad (2)$$

$$C_{max,i} + E_i - T_i = D_i \qquad \forall i \quad (3)$$

$$\sum_{s=1}^{S} Y_{i,s} = 1 \qquad \forall i \quad (4)$$

$$C_{i,m} \geq p_{i,m} \qquad \forall i, m \quad (5)$$

$$C_{i,m} \geq C_{i,n} + p_{i,m} \qquad \forall i, m, n \neq m | a_{i,m,n} = 1 \quad (6)$$

$$C_{j,m} \geq C_{i,m} + p_{j,m} - L(3 - X_{i,j,m} - Y_{i,s} - Y_{j,s}) \quad \forall m, s, i < J, j > i \quad (7)$$

$$C_{i,m} \geq C_{j,m} + p_{i,m} - L(2 + X_{i,j,m} - Y_{i,s} - Y_{j,s}) \quad \forall m, s, i < J, j > i \quad (8)$$

$$Y_{i,s} \in \{0,\ 1\} \qquad \forall i, s \quad (9)$$

$$X_{i,j,m} \in \{0,\ 1\} \qquad \forall i < J, j > i, m \quad (10)$$

$$C_{i,m} \geq 0; \quad E_i \geq 0; \quad T_i \geq 0 \qquad \forall i, m \quad (11)$$

Equation (1) is the objective function. The constraint (2) specifies that $C_{max,i}$ is the latest completion time of one of the individual processing steps on one of the machines. The earliness or tardiness of each job is calculated in (3). Then, (4) ensures that each job can only be assigned to one factory. Constraint (5) determines that at least the corresponding processing time must have elapsed before a job can be transferred to the next machine. The following machining step can be terminated at the earliest when the production time for this step has elapsed, in addition to all previous processing times (6). Constraint sets (7) and (8) indicate that a machine can only process one job at a time: If two orders $i$ and $j$ are processed in the same factory and $j$ is processed on machine $m$ after $i$, then $j$ must be completed at least by its processing time $p$ on machine $m$ later than $i$. Constraint sets (9), (10) and (11) specify the decision variables of the model.

The model can be extended to focus on minimizing costs of due date deviation (early or late completion). If a specific weighting factor can be assigned to each customer order (indicating the importance of different customer orders), the model (with the remaining parameters and decision variables unchanged) can be adapted to:

| | |
|---|---|
| $w_{E,i}$ | Weighting factor for earliness of job $i$ |
| $w_{T,i}$ | Weighting factor for tardiness of job $i$ |

$$Minimize\ V_g = \sum_{i=1}^{J} w_{T,i} \cdot T_i + \sum_{i=1}^{J} w_{E,i} \cdot E_i \quad (12)$$

As the determination of the weighting factors is very subjective, we focus on the first model without weighting factors in this contribution. Nevertheless, the adaption highlights the flexibility and simplicity of the developed MILP model.

## III. HEURISTICS AND META-HEURISTICS

### A. Heuristics (Two-stage iterative approaches)

One way to reduce the complexity of the problem is by taking an iterative solution approach. The two decisions in the DJSP are split into two phases to find near-optimal solutions in reasonable computation times. First, jobs are assigned to factories, followed by the sequencing of jobs on machines. Here, the assignment rule significantly determines the quality of the solution. Allocating the jobs according to descending due dates does not lead to a promising solution. Alternatively, the jobs can be sorted according to their sum of processing times. The difference between this makespan and the due date ("slack") is then used to determine the assignment sequence. In this way, the flexibility in planning is enhanced by preferring urgent jobs - at the expense of longer idle times for jobs with a lot of slack - when creating the plan. However, as the machines are still unscheduled when jobs are assigned, measuring the specific earliness of jobs is not trivial. Therefore, we solely focus on the tardiness of jobs for the used assignment rule, which is conceptually similar to the NEH2 rule [17] that performed well for the distributed permutation flow shop scheduling problem. The main difference is the order in which the jobs are selected, which is minimum slack, i.e., the buffer time. Then, the first $s$ orders are assigned to the $s$ plants one by one in order to separate the most urgent jobs and avoid interference. Each factory receives one urgent job first to ensure that the urgent jobs are started immediately. Subsequently, for the next job in the sorted list, the assignment that leads to the earliest completion time is computed. Various priority/dispatching rules may be used for this calculation. However, rules like FIFO/FCFS or SPT do not yield good results in the context of due date consideration. We focus on rules that include slack and can be used intuitively to meet due dates:

- Minimum slack (MSLACK): MSLACK iteratively checks the slack of all jobs and selects the job with the least slack whenever a machine is free. This includes observing the other restrictions like the machine sequence of each job.
- Slack per remaining processing time (S/RPT): S/RPT divides the remaining slack by the sum of outstanding raw processing times, to avoid the tendency of MSLACK to prefer long processing steps over shorter ones. The jobs are then sorted in ascending order of this quotient and assigned accordingly.
- Slack per remaining operations (S/OPN): Another way to modify MSLACK is by integrating the number of remaining operations in the calculation. Jobs are then sorted according to the remaining slack for each remaining process step.

The following steps are similar for all heuristics. Each job is assigned to a factory and the resulting completion times determined. Once all jobs are assigned, the sequence for each factory is determined, either by again using the three presented rules or by using more sophisticated approaches like the GH1 heuristic [12] presented next.

## B. Two-stage greedy heuristic (GH1)

Greedy heuristics have been successfully applied to many combinatorial problems and usually derive a performant solution in a reasonable amount of time. For the problem at hand, we use an encoding scheme for GH1 that consists of as many permutations as factories are available. Each value in the permutation represents an operation and each job appears as often as it needs operations processing. As the quality of the algorithm depends on a performant starting solution, the start permutation is randomly mixed. Subsequently, the operations are iteratively inserted into the sequence of operations in each factory. Naderi and Azab [12] use an additional rule for job-factory assignments based on the smoothing of workload and consider each machine separately. We use GH1 with the MSLACK rule in this article to focus on the minimization of due date deviations.

## C. Greedy heuristic (integrated approach)

So far, the presented heuristics and rules focused on a two-stage approach. We now turn to an integrated approach of assigning jobs to factories and sequencing jobs on machines in each factory. A greedy heuristic (GH3 by [12]) was adapted to the examined problem with a due date focus. The job-factory assignment is based on minimum slack and the shortest cycle times in the plants. The processing sequences of the jobs are calculated in each iteration using the GH1 procedure, where all plants are considered in each step. The procedure of the algorithm is as follows:

First, one job is assigned to each factory. Again, jobs with minimum slack are selected first and the makespan is calculated based on their processing times. The next job is then assigned to the factory with the shortest makespan after the assignment of the respective job. This is done by adding each operation to the previous sequence and inserting it in all possible positions, without, however, changing the previously determined sequence. For each position in the sequence, the makespan is recorded and then compared with the other position to select the best solution. If several positions lead to the same (best) solution, the earliest is chosen. This selection enables the jobs that are scheduled subsequently to be completed as early as possible and potential delays can be avoided. Unlike the two-stage algorithm GH1, GH3 does not depend on a random start sequence.

To specifically focus GH3 on due dates, we created a further adaption, denoted as GH3_SlackMin: The job-factory assignment is made after calculating the slack of all orders. Each job is iteratively assigned to each factory and scheduled using the exchange procedure. Then the remaining slack of all jobs is cumulated for the sequence. The job is assigned to the factory where the slack is maximized, that is, where flexibility is higher compared to the other locations. In this way, avoiding tardy orders is especially considered.

## D. Genetic algorithm adapted for the DJSP with due date objective

Genetic algorithms (GA) are popular meta-heuristics for scheduling problems due to their ability to investigate large solution spaces and simultaneously apply local search methods. We use a simple chromosome encoding for the adapted GA (based on [8]) aimed at the DJSP with a due date objective:

{'1j03' '2j01' '2j05' '1j04' '2j01' '1j02' '1j02' '1j03' '1j04' '2j05'}

The first number indicates the factory, followed by the job number. The first appearance of the job number indicates the first operation, the second appearance the second operation and so on. The fitness value that is calculated for each chromosome refers to the due date deviation of each developed schedule. Rather than randomly assigning a job to a factory during initialization and only changing this assignment rarely using global mutation, we modify the crossover and mutation operators to maintain a more extensive search space for the algorithm.

The initial job assignment is done separately for each chromosome in GA_adapted (see Fig. 2). In the iterations, two chromosomes are selected for the crossover process and the subsequences are exchanged. The feasibility of the new solution is checked and in case of a violation (i.e., job operations now spread among more than one factory), the crossover procedure overwrites the previous factory assignment for all further genes of this job. Thus, each generation explores new options for factory assignment using the crossover operator. Additionally, the best chromosome from the last iteration and the best chromosome overall are transferred to the next generation.

```
For z=1 to PopSize do
        Create pool of operations through random job assignment (Encoding)
        Generate chromosome through random permutation of operations
        Test chromosome for fitness
Endfor
Sort chromosomes of initial population according to fitness
For g=1 to NbGen do
        Select best chromosome as first chromosome for new generation
        Calculate selection probability through Linear Ranking Method
        Generate new population through Crossover and Mutation
        Sort chromosomes of new generation according to fitness
        If no new best chromosome is found for 20 consecutive operations
        Break
        Endif
Endfor
```

Fig. 2. The general outline of GA_adapted.

The succeeding chromosome is assessed regarding its fitness value after crossover and after mutation. The old chromosome is replaced if the new one outperforms it. This facilitates the movement of chromosomes towards a better solution and the crossover procedure prevents an excessive restriction of the search space.

In mutation, a better chromosome is taken over from the next generation with a certain probability. The selection probability is determined using a ranking of chromosomes based on their fitness value that is then transformed into a vector of selection probabilities ($p$) using the following equation [18]:

$$p_{LinRank,i} = \frac{\alpha + \left(\frac{rank_i}{(PopSize - 1)}\right) \times (\beta - \alpha)}{PopSize}$$

$$\text{with } \alpha = 2 - \beta; \quad 0 \leq \alpha \leq 1, 1 \leq \beta \leq 2$$

The greater $\beta$ is, the greater the probability of selecting the best chromosome. If $\alpha = 0$, the worst chromosome is never selected for mutation or crossover. For $\alpha = 0.01$, the best chromosome is twice as likely to be selected than the median, which does not restrict the search space too much as all chromosomes might be selected. All steps in the iteration procedure occur for a fixed set of iterations and the algorithm is terminated if no improvement is determined for several iterations. The parameters of the GA were determined using several small instances of the MILP model and led to the following specifications:

TABLE I
PARAMETERS OF THE GENETIC ALGORITHM (GA_ADAPTED)

| | |
|---|---|
| Population size | 300 |
| Number of generations | 100 |
| Termination criterion | 20 iterations without improvements |
| α | 0.01 |
| β | 1.99 |
| Crossover probability | 0.9 |
| Mutation probability | 0.9 |

## IV. EXPERIMENTAL EVALUATION AND DISCUSSION

In this section, we use conducted experiments to evaluate the performance of the model and the proposed heuristics. We start with the evaluation of the model on small instances before we move on to larger instances to compare the performance of the heuristics. The model was formulated and solved using CPLEX 12.10, whereas the algorithms were coded in MATLAB. All runs were conducted with a PC with 1.6 GHz i5-8250 CPU and 8 GB of RAM. The performance measure used in this contribution is the relative percentage deviation (RPD), with *Alg* as an indicator for the objective value of the tested algorithm and *Min* as the best solution that was obtained by any algorithm for the given problem instance:

$$RPD = \frac{Alg - Min}{Min} \times 100$$

So far, there are no benchmark sets for the DJSP taking due dates into account. Therefore, we created some smaller instances to test the MILP model. Parameters include jobs (J), machines (M), factories (S) and the processing time $p_{i,m}$, which was randomly set to U (1,99). Setting realistic due dates has a significant impact on the quality of the solution. We experimented with this parameter and set a value of 120% of the sum of processing times of each job for the tested small instances. For the larger instances, this value was increased to a maximum of 200% to take into account the increasing number of orders and potentially more scheduling conflicts. The time limit for the model and the algorithms was set at 3600 seconds.

Table II indicates the results of the MILP model for the small instances. The model is able to solve the smaller instances to optimality, whereas for the larger models, CPLEX fails to determine the optimum solution (at least in the case of 2 factories). Regarding the complexity of the model, it is quadratic in *J*, but linear in *M* and *S*. For example, for the instance 10x4x2, the number of constraints is 870. The variables amount to 200 binary, 70 integers, and 60 continuous. The most influential parameter on the complexity of the model is the number of jobs *J*.

TABLE II
RESULTS OF THE MILP MODEL ON SMALL INSTANCES

| J | M | S | Model | | |
|---|---|---|---|---|---|
| | | | V | Time (s) | Opt. Gap (%) |
| 6 | 3 | 2 | **8** | 0,11 | 0 |
| 6 | 4 | 2 | **49** | 0,30 | 0 |
| 8 | 3 | 2 | **64** | 0,41 | 0 |
| 8 | 4 | 2 | 1723 | 4,22 | 12,5 |
| 10 | 3 | 2 | 435 | 21,11 | 19,7 |
| 10 | 4 | 2 | 241 | 14,23 | 43,9 |
| 10 | 3 | 3 | **68** | 1,08 | 0 |
| 10 | 4 | 3 | 59 | 1,36 | 89,8 |
| 12 | 3 | 2 | 548 | 329,19 | 1,6 |
| 12 | 4 | 2 | 371 | 53,72 | 14,5 |
| 12 | 3 | 3 | 128 | 6,36 | 16,4 |
| 12 | 4 | 3 | **40** | 3,03 | 0 |

TABLE III
OBJECTIVE VALUES OF THE ALGORITHMS ON LARGER INSTANCES

| **Inst.** | **Algorithms** | | | | | |
|---|---|---|---|---|---|---|
| | MSLACK | S/RPT, S/OPN | GH1_MSLACK | GH3 | GH3_SlackMin | GA_adapted |
| 10x10 | 2453 | 2453 | 2160 | 2278 | 2085 | **1754** |
| ft06 | 35 | 35 | 29 | 44 | 42 | **8** |
| ft06 | 11 | 11 | 9 | 8 | 8 | **5** |
| ft10 | 1537 | 1537 | 1687 | 1379 | 2204 | **1247** |
| ft10 | 562 | 562 | 686 | 962 | 549 | **473** |
| ft10 | 585 | 585 | 662 | 700 | 520 | **215** |
| ft20 | 1886 | 1886 | 1608 | 1227 | **756** | 1044 |
| tai15 | 1840 | 1540 | 1732 | **1473** | 2010 | 2903 |
| tai15 | 1545 | 1930 | 1226 | 1947 | **984** | 987 |
| tai50 | 10198 | 12274 | 13935 | infs. | infs. | **9558** |
| Average RPD | 95% | 99% | 85% | 116% | 86% | 15% |

All the heuristics were able to solve the small instances in ≤1 s, mostly with the optimal solution. Therefore, and due to space restrictions, we only present the values of the MILP model in Table II and do not present the heuristics. To evaluate the heuristics and meta-heuristics for larger instances, we derived a second set of ten instances based on the benchmark instances of Taillard [15], denoted as "tai" in Table III, as well as Fisher and Thompson [16], denoted as "ft". The results of the best performing heuristics and the meta-heuristic are displayed in Table III. The values in bold represent the best performance value found (*Min*). Table IV indicates the different instances with the parameter values used. Here, DDO represents the due date offset used.

TABLE IV
THE INSTANCES USED FOR THE HEURISTICS EVALUATION

| **Instance** | **J** | **M** | **S** | **DDO** |
|---|---|---|---|---|
| 10x10 | 10 | 10 | 2 | 500 time units |
| ft06 | 6 | 6 | 2 | 120% |
| ft06 | 6 | 6 | 3 | 120% |
| ft10 | 10 | 10 | 2 | 120% |
| ft10 | 10 | 10 | 3 | 120% |
| ft10 | 10 | 10 | 4 | 120% |
| ft20 | 20 | 5 | 5 | 150% |
| tai15 | 15 | 15 | 2 | 150% |
| tai15 | 15 | 15 | 3 | 150% |
| tai50 | 50 | 15 | 5 | 200% |

The genetic algorithm GA_adapted performed best overall and was able to determine the best solution in 7 of 10 instances. The total RPD for GA_adapted is also the lowest at 15%. Due to the high number of generations, the computational times of GA_adapted are also the longest on average. The second best performing heuristic is GH1_MSLACK, which was run for 20 times for each problem instance to generate different random start sequences and assess the solution quality comprehensively.

If we specifically compare the two-stage algorithm GH1_MSLACK with the integrated algorithms GH3 and GH3_SlackMin, we find that for medium problem sizes the integrated approach does not perform better than the two-stage approach. The integrated approaches only outperform the two-stage approach on larger instances with more than ten orders where the advantage of iterative assignment to factories and sequencing is exploited.

The simple heuristic rules MSLACK, S/RPT, and S/OPN show promising results in very short computation times $(< 1\,s)$. Interestingly, they even outperform the greedy heuristics in some instances. This may be caused by the due date objective, which is explicitly considered in the three rules. The greedy heuristics were originally designed to minimize the makespan and therefore arrange jobs to be completed as early as possible, which might lead to earliness and thus to a deterioration of the objective value.

Further adjustments or new algorithm designs seem promising in this context. Other metaheuristics could also be meaningfully applied to the DJSP. For example, adaptations of simulated annealing or iterated greedy heuristics have performed well on related problem settings and may be considered [19], [20]. The need for standardized benchmark instances for the DJSP with a focus on due date related objectives has also become apparent.

## V. CONCLUSION

Research on distributed scheduling in a job shop configuration that also considers due dates is scarce. However, ensuring on-time delivery is becoming increasingly important for several manufacturing industries and there is a need for models and algorithms that focus on due dates. In this contribution, we developed and presented a MILP model for the DJSP with a due date objective. Several heuristics from the single-factory scheduling literature were adapted to the specific problem at hand. Additionally, a genetic algorithm was presented to solve larger instances of the DJSP effectively. The scheduling model and the algorithms can be readily applied in real-world settings, where manufacturing is distributed among several factories. In this way, industries can advance towards modern configurations of manufacturing systems. The experimental results indicate that the heuristics and metaheuristics provide promising solutions in reasonable computation times. They are better suited for large problem instances than the exact model. Future research should deal with heterogeneous factories, where different machine types or processing times are present in the factories as well as stochastic and dynamic demand. Besides, additional characteristics should be considered in the model to make the problem setting more realistic.

This could also include multi-objective problems, as many practical settings in the industry are subject to multiple objectives. For instance, it might be worth striving to find a balance between efficient manufacturing with a high machine utilization, focusing on makespan, and the fulfillment of customer requested delivery dates, e.g. tardiness.

## ACKNOWLEDGMENT

*This research was supported by the EU project iDev40. The project iDev40 has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 783163. The JU receives support from the European Union's Horizon 2020 research and innovation program. It is co-funded by the consortium members, as well as grants from Austria, Germany, Belgium, Italy, Spain and Romania. The content of this article does not reflect the official opinion of the Joint Undertaking ECSEL. Responsibility for the information and views expressed in the article lies entirely with the authors.*

## REFERENCES

[1] J. Olhager and A. Feldmann, 'Distribution of manufacturing strategy decision-making in multi-plant networks', *Int. J. Prod. Res.*, vol. 56, no. 1–2, pp. 692–708, 2018, DOI.10.1080/00207543.2017.1401749.

[2] J. Behnamian and S. M. T. Fatemi Ghomi, 'A survey of multi-factory scheduling', *J. Intell. Manuf.*, vol. 27, no. 1, pp. 231–249, 2016, DOI.10.1007/s10845-014-0890-y.

[3] R. Ruiz, Q.-K. Pan, and B. Naderi, 'Iterated Greedy methods for the distributed permutation flowshop scheduling problem', *Omega*, vol. 83, pp. 213–222, Mar. 2019, DOI.10.1016/j.omega.2018.03.004.

[4] Z. X. Guo, W. K. Wong, Z. Li, and P. Ren, 'Modeling and Pareto optimization of multi-objective order scheduling problems in production planning', *Comput. Ind. Eng.*, vol. 64, no. 4, pp. 972–986, 2013, DOI.10.1016/j.cie.2013.01.006.

[5] T.-K. Liu, Y.-P. Chen, and J.-H. Chou, 'Solving Distributed and Flexible Job-Shop Scheduling Problems for a Real-World Fastener Manufacturer', *IEEE Access*, vol. 2, pp. 1598–1606, 2015, DOI.10.1109/access.2015.2388486.

[6] J. R. Jackson, 'An extension of Johnson's results on job IDT scheduling', *Nav. Res. Logist. Q.*, vol. 3, no. 3, pp. 201–203, Sep. 1956, DOI.10.1002/nav.3800030307.

[7] J. Lohmer and R. Lasch, 'Production planning and scheduling in multi-factory production networks: a systematic literature review', *Int. J. Prod. Res.*, pp. 1–27, Jul. 2020, DOI.10.1080/00207543.2020.1797207.

[8] H. Z. Jia, A. Y. C. Nee, J. Y. H. Fuh, and Y. F. Zhang, 'A modified genetic algorithm for distributed scheduling problems', *J. Intell. Manuf.*, vol. 14, no. 3–4, pp. 351–362, 2003, DOI.10.1023/A:1024653810491.

[9] F. T. S. Chan, S. H. Chung, and P. L. Y. Chan, 'Application of genetic algorithms with dominant genes in a distributed scheduling problem in flexible manufacturing systems', *Int. J. Prod. Res.*, vol. 44, no. 3, pp. 523–543, 2006, DOI.10.1080/00207540500319229.

[10] L. De Giovanni and F. Pezzella, 'An Improved Genetic Algorithm for the Distributed and Flexible Job-shop Scheduling problem', *Eur. J. Oper. Res.*, vol. 200, no. 2, pp. 395–408, 2010, DOI.10.1016/j.ejor.2009.01.008.

[11] F. T. S. Chan, A. Prakash, H. L. Ma, and C. S. Wong, 'A hybrid Tabu sample-sort simulated annealing approach for solving distributed scheduling problem', *Int. J. Prod. Res.*, vol. 51, no. 9, pp. 2602–2619, 2013, DOI.10.1080/00207543.2012.737948.

[12] B. Naderi and A. Azab, 'Modeling and heuristics for scheduling of distributed job shops', *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7754–7763, 2014, DOI.10.1016/j.eswa.2014.06.023.

[13] M. C. Wu, C. S. Lin, C. H. Lin, and C. F. Chen, 'Effects of different chromosome representations in developing genetic algorithms to solve DFJS scheduling problems', *Comput. Oper. Res.*, vol. 80, pp. 101–112, 2017, DOI.10.1016/j.cor.2016.11.021.

[14] Z. X. Guo, C. Yang, W. Wang, and J. Yang, 'Harmony search-based multi-objective optimization model for multi-site order planning with multiple uncertainties and learning effects', *Comput. Ind. Eng.*, vol. 83, pp. 74–90, 2015, DOI.10.1016/j.cie.2015.01.023.

[15] E. Taillard, 'Benchmarks for basic scheduling problems', *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, Jan. 1993, DOI.10.1016/0377-2217(93)90182-M.

[16] R. Fisher and G. Thompson, 'Probabilistic Learning Combinations of Local Job-shop Scheduling Rules', *Ind. Sched.*, pp. 225–251, 1963.

[17] B. Naderi and R. Ruiz, 'The distributed permutation flowshop scheduling problem', *Comput. Oper. Res.*, vol. 37, no. 4, pp. 754–768, 2010, DOI.10.1016/j.cor.2009.06.019.

[18] J. Grefenstette, 'Rank-based selection', in *Handbook of Evolutionary Computation*, 1st editio., T. Back, D. B. Fogel, and Z. Michalewicz, Eds. Oxford, UK: Oxford Univ. Press, 1997, p. Chap C2.4.

[19] V. Fernandez-Viagas and J. M. Framinan, 'A bounded-search iterated greedy algorithm for the distributed permutation flowshop scheduling problem', *Int. J. Prod. Res.*, vol. 53, no. 4, pp. 1111–1123, 2015, DOI.10.1080/00207543.2014.948578.

[20] Q.-K. Pan, L. Gao, L. Xin-Yu, and J. M. Framinan, 'Effective constructive heuristics and meta-heuristics for the distributed assembly permutation flowshop scheduling problem', *Appl. Soft Comput.*, vol. 81, p. 105492, 2019, DOI.10.1016/j.asoc.2019.105492.