CHEMICAL ENGINEERING TRANSACTIONS
VOL. 121, 2025
A publication of AIDIC
The Italian Association of Chemical Engineering
Online at www.cetjournal.it
Guest Editors: Petar S. Varbanov

ISBN 979-12-81206-22-9; **ISSN** 2283-9216
DOI: 10.3303/CET25121010

Sustainability in Manufacturing: MILP Models of a Production Line Optimization Problem

Tamás Hajba

Department of Mathematics and Physics, Széchenyi István University, Egyetem tér 1, 9026, Győr, Hungary
hajbat@math.sze.hu

In modern factories, the optimization of production lines is essential to reduce energy consumption and costs. A well-known model of production line optimization is the Permutation Flow Shop Problem (PFSP) with makespan minimization. PFSP instances arising in industry often have some special features: the problem may contain jobs whose processing times are equal on every machine. Such jobs are said to be of the same type. Due to technological reasons, some of the permutations may be forbidden. For example, only those job permutations are allowed in which, for each job type, the length of a maximal block of jobs from that type is divisible by a prescribed number (the lot size of that type). Considering these features, this paper introduces a generalization of the Permutation Flow Shop with Repetition and Lot Size Problem, in which different types of jobs have different lot sizes. We present two new mixed integer linear programming formulations of the problem and compare their effectiveness on a set of benchmark problems. Both models found the optimal solution within a few seconds for all problem instances involving 10 machines, 100 jobs, and five different types. These results show that even large-scale problems containing a small number of different jobs can be solved using these models.

1. Introduction

Flow shop scheduling is frequently used in real-life manufacturing systems, for example, at factories in the automotive and semiconductor industries. Since the best schedule of the jobs can significantly decrease the operational costs and energy consumption, companies use some optimization software to find a near-optimal solution within a restricted computational time.

The Permutation Flow Shop Problem (PFSP) with makespan minimization is a well-known combinatorial optimization problem in which a set of jobs has to be processed on a set of machines. Each job has to visit each of the machines in the same order, and the order of the jobs has to be the same on each machine. Solution methods for the PFSP can be divided into 4 categories: exact algorithms, heuristics, metaheuristics, and methods using artificial intelligence.

Exact methods are capable of finding the optimal solution for small or medium-sized problems. Gmys et al. (2020) gave an efficient branch-and-bound algorithm for the PFSP. Ozolin (2019) proposed a dynamic programming algorithm to solve PFSPs in which there is a limited buffer size between consecutive machines. Tseng and Stafford (2008) introduced mixed integer linear programming (MILP) models for the PFSP and compared them on a set of instances.

Among the heuristics, the most well-known and cited is the NEH heuristic introduced by Nawaz et al. (1983). Later, the original NEH was improved by several authors, for example, by Zhang et al. (2023). Zhao et al. (2021) applied iterated greedy algorithms to solve the PFSP. A review of heuristics for the PFSP can be found in Ruiz and Maroto (2005).

As for the metaheuristics, Tseng and Lin (2009) combined a genetic algorithm and local search to find a near-optimal solution of the PFSP. Tasgetiren et al. (2004) introduced a particle swarm optimization algorithm to minimize the makespan, while Kurdi (2022) proposed a novel ant colony optimization (ACO) algorithm for the open shop scheduling problem. Chen and Zheng (2024) applied a hybrid grey wolf optimization method to solve the PFSP.

Paper Received: 15 July 2025; Revised: 29 August 2025; Accepted: 24 October 2025
Please cite this article as: Hajba T., 2025, Sustainability in Manufacturing: MILP Models of a Production Line Optimization Problem, Chemical Engineering Transactions, 121, 55-60 DOI:10.3303/CET25121010

Recently, machine learning algorithms have been applied to solve combinatorial optimization problems. Pan et al. (2021) introduced an algorithm based on deep reinforcement learning for the PFSP. Brammer et al. (2022) used reinforcement learning to solve the PFSP with multiple lines and demand plans. Zhao et al. (2023) integrated Q-learning into scatter search to tackle the distributed permutation flow shop problem with sequence-dependent setup times. Guo et al. (2025) combined the NEH algorithm and Q-learning for the classical PFSP.
The advantage of solving the mixed integer linear programming (MILP) formulation of the PFSP via some numerical solver (such as CPLEX) is that this approach guarantees an optimal solution. But on the other hand, since the PFSP is an NP-hard problem if the number of machines is at least 3, the drawback of this approach is that we cannot expect that large-scale problems (containing a lot of jobs or machines) can be solved this way. However, the continuous progress in computers and numerical solvers implies that larger and larger problems can be solved by applying this method. Real-life PFSPs may contain some special features that may yield MILP models that are easier for the solvers to solve.
First, in many industrial problems, jobs can be categorized into different types. Two jobs have the same type if their processing times are the same on any machine. The corresponding Permutation with Repetition Flow Shop The problem and its MILP models were introduced in Hajba and Horváth (2013). Secondly, in real-life applications, some permutations of the jobs are prohibited. For example, only those permutations are permitted in which for each type i, the size of every maximal subsequence of jobs of type i is divisible by a prescribed number s_i (which is called the lot size of i). This problem, with the additional constraint that each type has the same lot size, called the Permutation with Repetition Flow Shop with Lot Size (RL-PFSP), was examined in Hajba and Horváth (2013). In this article, the generalization of this problem, in which different types may have different lot sizes, is studied. The corresponding problem will be called the Permutation Flow Shop with Repetition and Different Lot Sizes Problem (RDL-PFSP). Although the RDL-PFSP can be considered as a special case of the group scheduling problem, which was studied in several papers, for example, in (Yuan et al., 2021), this special case of the problem has not yet been investigated.
The rest of the paper is organized as follows. The proper definition of the problem is given in Section 2. In Section 3, two new MILP models of the RDL-PFSP are introduced. The effectiveness of the MILP models is tested on a set of benchmark problems. The results of the numerical experiments are presented in Section 4. Finally, Section 5 contains the conclusions.

2. Definition of the RDL-PFSP

The RDL-PFSP can be defined as follows:

- A set of n jobs has to be processed on a set of m machines.
- Each job has to be processed first on the first machine, then on the second machine, and so on.
- The sequence of the jobs has to be the same on every machine.
- A machine can process at most one job at a time.
- Interruption of the processing of a job is not allowed.
- Each job is available for processing at time 0.
- There is an unlimited buffer between any two consecutive machines.
- For each job and machine pair, the processing time of the job on the machine is known.
- The jobs are classified into different types. If two jobs have the same type, they have the same processing time on every machine.
- For every type i, the size of each maximal subsequence of jobs of type i is divisible by s_i, which is the lot size of type i.
- The goal is to minimize the completion time of the last job of the sequence on the last machine.

The unlimited buffer between consecutive machines implies that a job can immediately leave a machine as soon as its processing on the machine is finished. In contrast, in the case of a limited buffer size between consecutive machines, a job cannot leave a machine after its processing is finished if the buffer is full. These problems are more complex and yield a larger optimal value. To illustrate the meaning of the lot size, let us suppose that we have a problem with 10 jobs and two types. There are six jobs of the first type and four jobs of the second type. The lot size of the first type is 3, and the lot size of the second type is 2. Let us denote jobs of the first type by 1 and jobs of the second type by 2. In this case, the permutation 1112221112 is not a feasible solution because it contains a maximal subsequence of jobs of type 2 whose size is 3, which is not divisible by the lot size of the second type (which is 2). Similarly, the permutation 2211111122 is a feasible permutation. Every feasible permutation can be divided into subsequences of jobs such that each subsequence contains jobs from the same type and the size of the subsequence equals the lot size of the type. Such a subsequence is called a lot. Lot sizes can occur, for example, in industrial problems where jobs of the same type are transported to the beginning

of the assembly line in a special container, and the jobs carried in the same container need to be placed on the line one after another.

3. MILP models of the RDL-PFSP

In this section, two new MILP models of the RDL-PFSP are formulated. The new models are based on the MILP models of the R-FPSP that were introduced in Hajba and Horváth (2013).
In order to describe a permutation, the type of job placed at each position i in the sequence must be specified. To do so, it is sufficient to give the starting position of each lot. For example, if the lot size of type i is 3, and a lot containing jobs of type i starts at the ninth position of the sequence, then the ninth, tenth, and eleventh positions in the sequence will contain jobs of type i. For each (r,j) pair, the start time (or equivalently the completion time) of processing the j-th job in the sequence on machine r must be determined. Taking these into consideration, the following notations are used in the MILP models:

Parameters:

M – number of machines
T – number of different types of jobs
L_i – number of lots containing jobs of type i ($1 \le i \le T$)
s_i – size of a lot containing jobs of type i ($1 \le i \le T$)
N – total number of jobs
$P_{r,i}$ – processing time of job of type i on machine r ($1 \le r \le M$, $1 \le i \le N$)

Continuous variables:

$B_{r,j}$ – beginning time of processing the j-th job of the sequence on machine r ($1 \le r \le M$, $1 \le j \le N$)
$C_{r,j}$ – completion time of processing the j-th job of the sequence on machine r ($1 \le r \le M$, $1 \le j \le N$)
C_{max} – makespan

Binary Variables:

$Z_{i,j}$ – equals 1 iff a lot containing jobs of type i starts at the j-th position of the sequence ($1 \le i \le T$, $1 \le j \le N$)

3.1 The RDL-TS2 model:

$$\sum_{j=1}^{N} Z_{i,j} = L_i\,,\ \ 1 \le i \le T \tag{1}$$

$$\sum_{i=1}^{T} \sum_{k=j-s_i+1}^{j} Z_{i,k} = 1\,,\ \ 1 \le j \le N \tag{2}$$

$$C_{r,j} + \sum_{i=1}^{T} \sum_{k=j-s_i+2}^{j+1} Z_{i,k} \cdot P_{r,i} \le C_{r,j+1}\ ,\ \ 1 \le r \le M,\ 1 \le j \le N-1 \tag{3}$$

$$C_{r,j} + \sum_{i=1}^{T} \sum_{k=j-s_i+1}^{j} Z_{i,k} \cdot P_{r+1,i} \le C_{r+1,j}\ ,\ \ 1 \le r \le M-1, 1 \le j \le N \tag{4}$$

$$\sum_{i=1}^{T} Z_{i,1} \cdot P_{1,i} \le C_{1,1} \tag{5}$$

$$Z_{i,j} = 0\,,\ \ 1 \le i \le T\,, N - s_i + 2 \le j \le N \tag{6}$$

$$C_{max} = C_{M,N} \to min \tag{7}$$

Equation (1) says that there are exactly L_i lots containing jobs of type i. Constraint (2) ensures that each job in the sequence is contained in exactly one lot. Constraint (3) states that the (j+1)-th job of the sequence can only be finished on any machine after the j-th job of the sequence is finished on the same machine and the (j+1)-th job of the sequence is processed on the same machine. Constraint (4) ensures that a job can be finished on any machine only after it is finished on the previous machine and processed on the current machine. Constraint (5) states that completion time of the first job of the sequence on the first machine cannot be less than the processing time of the first job of the sequence on the first machine. Constraint (6) ensures that for each type i, a lot cannot start at the j-th position of the sequence if there are at most s_i-2 positions after j. Constraint (7) ensures that the completion time of the last job of the sequence on the last machine (i.e., the makespan) has to be minimized.

3.2 The RDL-Wilson model:

$$\sum_{j=1}^{N} Z_{i,j} = L_i\ ,\ \ 1 \le i \le T \tag{8}$$

$$\sum_{i=1}^{T} \sum_{k=j-s_i+1}^{j} Z_{i,k} = 1\ ,\ \ 1 \le j \le N \tag{9}$$

$$B_{r,j} + \sum_{i=1}^{T} \sum_{k=j-s_i+1}^{j} Z_{i,k} \cdot P_{r,i} \le B_{r,j+1}\ ,\ \ 1 \le r \le M, 1 \le j \le N-1 \tag{10}$$

$$B_{r,j} + \sum_{i=1}^{T} \sum_{k=j-s_i+1}^{j} Z_{i,k} \cdot P_{r,i} \le B_{r+1,j}\ ,\ \ 1 \le r \le M-1, 1 \le j \le N \tag{11}$$

$$Z_{i,j} = 0\ ,\ \ 1 \le i \le T\ , N - n_i + 2 \le j \le N \tag{12}$$

$$B_{1,1} = 0 \tag{13}$$

$$C_{max} = B_{M,N} + \sum_{i=1}^{T} \sum_{j=N-s_i+1}^{N} Z_{i,j} \cdot P_{M,i} \to min \tag{14}$$

As in the RDL-TS2-model, equation (8) states that there are exactly L_i lots containing jobs of type I, while constraint (9) ensures that each job in the sequence is contained in exactly one lot. Constraint (10) states that the processing of the (j+1)-th job of the sequence can begin on any machine only after the j-th job of the sequence is finished on the same machine. Constraint (11) states that the processing of a job on any machine can be started only after the processing of that job is finished on the previous machine. Constraint (12) ensures that for each type i a lot cannot start at the j-th position of the sequence if there are at most s_i-2 positions after j. Equality (13) implies that the processing of the first job of the sequence starts on the first machine at time 0. Finally, (14) ensures that the completion time of the last job of the sequence on the last machine (i.e. the makespan) is minimized.

4. Numerical experiments

To compare the presented MILP models, a 4-cell experimental design, $M \in \{10,15\}$ and $N \in \{80,100\}$ was applied. Each cell contained five problems, so the MILP models were tested on a total of 20 problems. As in the well-known benchmark of Taillard (1993) for flow shop problems, the processing times of the jobs were random integers chosen from the uniform distribution in the range [1,100]. There were five different types in each example, i.e., T equaled 5 in each problem. The number of jobs was the same for each type, so in the problems with 100 jobs, there were 20 jobs of each type, while in the problems with 80 jobs, there were 16 jobs from each type. The lot sizes of the different types were (4,8,4,2,8) in the problems with 80 jobs, while they were (2,4,5,10,4) in the problems with 100 jobs. Each problem contained 20 lots.
The MILP models were implemented in GAMS (Bussieck (2004)) and solved by CPLEX 12.5 on an Intel Xeon E3 1225 3.1 GHz computer equipped with 4 GB of RAM. In addition to the default CPLEX options, the deterministic parallel mode with 2 threads and a time limit of 600 seconds was used.

4.1 Comparison of the models on problems with 10 machines

First, the RDL-TS2 and RDL-Wilson models were compared on 10 problems, each with 10 machines. For a given instance number, the problem with 80 jobs had the same processing time matrix as the corresponding problem with 100 jobs. Table 1 shows the solution times in seconds.

Table 1: Solution times (in sec) of the models for problems with M=10

	instance	RDL-TS2	RDL-Wilson
N=80	1	1.2	1.4
N=80	2	20.4	15.2
N=80	3	1.2	0.9
N=80	4	4.5	6.9
N=80	5	1.7	1.9
N=100	1	2	2.7
N=100	2	22.1	5.1
N=100	3	41	11.6
N=100	4	19.3	6.4
N=100	5	7.72	4.9

It can be seen that both models were able to find the optimal solution in at most 41 s. The RDL-Wilson model was faster in 6 out of the 10 instances. The mean solution time of the RDL-TS2 model was 5.6 seconds for problems with 80 jobs and 18.4 s for problems with 100 jobs. The mean solution time of the RDL-Wilson model was 5.2 seconds for problems with 80 jobs and 6.1 seconds for problems with 100 jobs. The Wilcoxon signed-rank test (Hollander, 2013) shows that there is no significant difference between the two models. However, if the three “easiest” instances, which were solved by both models in less than 2 s, are excluded from the comparison, the Wilcoxon signed-rank test shows that the RDL-Wilson model is significantly faster than the RDL-TS2 model. It can be concluded that the RDL-Wilson model performed better on the problems containing 10 machines than the RDL-TS2 model.

4.2 Comparison of the models on problems with 15 machines

Next, the two MILP models were tested on 10 problems, each with 15 machines. As in the previous section, for a given instance number, the problem with 80 jobs had the same processing time matrix as the corresponding problem with 100 jobs. Table 2 shows the solution times in seconds. The asterisk (*) in the table means that an optimal solution was not found within the 10 min runtime.

Table 2: Solution times (in sec) of the models for problems with M=15

	instance	RDL-TS2	RDL-Wilson
N=80	1	2.6	2.5
N=80	2	*	*
N=80	3	463	*
N=80	4	306	324
N=80	5	589	*
N=100	1	9.9	9.5
N=100	2	*	*
N=100	3	*	*
N=100	4	*	*
N=100	5	13.4	25.8

It can be seen that these instances were significantly more difficult for the MILP models than the instances with 10 machines, as the RDL-TS2 model failed to find the optimal solution within 10 min in 4 of the 10 instances, while the RDL-Wilson model failed to find the optimal solution within 10 min in 6 of the 10 instances. There were 4 problems that neither model could solve, and 2 problems that the RDL-TS2 model could solve, while the RDL-Wilson model could not solve. In those instances that both models could solve, the RDL-TS2 model was much faster than the RDL-Wilson model in 2 cases, and the RDL-Wilson model was slightly faster than the RDL-TS2 model in 2 cases. In those 4 problems that neither model could solve, the objective values of the best solutions found by the RDL-TS2 models were 6509, 8307, 7605, and 5255, while these values were 6504, 8304, 7600, and 5272 for the R-DL-Wilson model. In 3 out of the 4 cases, the RDL-Wilson model found a slightly better solution than the RDL-TS2 model. Summarizing these observations, it can be said that the RDL-TS2 model performed better on problems with 100 jobs than the RDL-Wilson model.

5. Conclusions

Optimization of production lines reduces a factory’s energy consumption, which yields a lower emission footprint for the company. Additionally, optimized lines extend the life cycle of machines, supporting sustainable equipment practices by delaying the need for replacement. Efficient lines can improve workers’ overtime conditions, enhancing overall human resource well-being. In this article, a special production line optimization problem, a generalization of the RL-PFSP, the Permutation Flow Shop with Repetition and Different Lot Size Problem (RDL-PFSP), is introduced. Two new mixed integer linear programming formulations, namely the RDL-TS2 model and the RDL-Wilson model, are introduced. To compare the effectiveness of the models, a benchmark set consisting of 20 problems was generated. The conclusions derived from the numerical experiments are as follows:

- Both models were able to find the optimal solution in less than 41 s in all of the 10 problems involving 10 machines.
- The RDL-Wilson model performed better on problems with 10 machines than the RDL-TS2 model.
- Problems with 15 machines were significantly more difficult for both models to solve.

- The RDL-TS2 model found the optimal solutions in 6 out of the 10 problems with 15 machines, while the RDL-TS2 model succeeded in 4 out of 10 such problems.

Production optimization software can integrate MILP solvers (such as CPLEX or GUROBI) by embedding them in the backend of applications to solve MILP models. In this way, our two models may be integrated with real production planning tools. With the rapid development of MILP solvers, we can expect increasingly larger industrial problems to be solvable using this approach.

In future work, we aim to investigate how the models can be improved in order to be able to solve larger problems. In this study, the default settings of CPLEX were used to solve the MILP models. It is also worth examining whether the modification of the settings of CLPEX could reduce the solution time.

Reference

Brammer J., Lutz B., Neumann D., 2022, Permutation flow shop scheduling with multiple lines and demand plans using reinforcement learning. European Journal of Operational Research, 299(1), 75-86.

Bussieck M.R., Meeraus A., 2004, General algebraic modeling system (GAMS). In: Modeling languages in mathematical optimization, Springer, Boston, MA, United States, 137-157.

Chen S., Zheng J., 2024, Hybrid grey wolf optimizer for solving permutation flow shop scheduling problem. Concurrency and Computation: Practice and Experience, 36(5), e7942.

Gmys J., Mezmaz M., Melab N., Tuyttens D., 2020, A computationally efficient Branch-and-Bound algorithm for the permutation flow shop scheduling problem. European Journal of Operational Research, 284(3), 814–833.

Guo D., Liu S., Ling S., Li M., Jiang Y., Li M., Huang G.Q., 2024, The marriage of operations research and reinforcement learning: Integration of NEH into Q-learning algorithm for the permutation flowshop scheduling problem. Expert Systems with Applications, 255, 124779.

Hajba T., Horváth Z., 2013, New effective MILP models for PFSPs arising from real applications. Central European Journal of Operations Research, 21, 729-744.

Hollander M., Wolfe D.A., Chicken E., 2013, Nonparametric statistical methods. John Wiley & Sons, Hoboken, NJ, United States, ISBN:9780470387375, DOI:10.1002/9781119196037.

Kurdi M., 2022, Ant colony optimization with a new exploratory heuristic information approach for open shop scheduling problem. Knowledge-Based Systems, 242, 108323.

Nawaz M., Enscore Jr E.E., 1983, A heuristic algorithm for the n-job flow-shop sequencing problem. Omega, 11(1), 91-95.

Ozolin A., 2019, Improved bounded dynamic programming algorithm for solving the blocking flow shop problem. Central European Journal of Operations Research, 27(1), 15-38.

Pan Z., Wang L., Wang J., Lu J., 2021, Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. IEEE Transactions on Emerging Topics in Computational Intelligence, 7(4), 983-994.

Ruiz R., Maroto C., 2005, A comprehensive review and evaluation of permutation flowshop heuristics. European Journal of Operational Research, 165(2), 479-494.

Taillard E., 1993, Benchmarks for basic scheduling problems. European Journal of Operational Research, 64(2), 278-285.

Tasgetiren M.F., Sevkli M., Liang Y.C., Gencyilmaz G., 2004, Particle swarm optimization algorithm for permutation flowshop sequencing problem. In: International Workshop on Ant Colony Optimization and Swarm Intelligence. Springer, Berlin/Heidelberg, Germany. 382-389.

Tseng F.T., Stafford Jr. E.F., 2008, New MILP models for the permutation flowshop problem. Journal of the Operational Research Society, 59(10), 1373-1386.

Tseng L.Y., Lin Y.T., 2009, A hybrid genetic local search algorithm for the permutation flowshop scheduling problem. European Journal of Operational Research, 198(1), 84-92.

Yuan S., Li T., Wang B., 2021, A discrete differential evolution algorithm for flow shop group scheduling problem with sequence-dependent setup and transportation times. Journal of Intelligent Manufacturing, 32, 427-439.

Zhang J., Dao S.D., Zhang W., Goh M., Yu G., Jin Y., Liu W., 2023, A new job priority rule for the NEH-based heuristic to minimize makespan in permutation flowshops. Engineering Optimization, 55(8), 1296-1315.

Zhao Z., Zhou M., Liu S., 2021, Iterated greedy algorithms for flow-shop scheduling problems. IEEE Transactions on Automatic Science and Engineering, 19(3), 1941-1959.

Zhao F., Zhou G., Wang L., 2023, A cooperative scatter search with reinforcement learning mechanism for the distributed permutation flowshop scheduling problem with sequence-dependent setup times. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 53(8), 4899-4911.