Supplementary Material: A Benchmarking Environment for Worker Flexibility in Flexible Job Shop Scheduling Problems

Authors[a]

[a]*Research Centre, Centre, University , Address, City, 0000, Country*

By describing three different problem representations for the FJSSP and the FJSSP-W instances, this supplementary section establishes the foundation to utilize the benchmarking environment for different types of solvers. This is intended to facilitate the comparison of algorithmic results with solvers from other optimization areas. In so doing, we do not claim to be universally valid and complete, but merely provide suggestions. The use of own and possibly more efficient problem formulations is not excluded, taking into account the benchmarking principles of reproducibility and comparability.

The following sections describe three types of problem formulations. The formulations are provided for the FJSSP instances, and for the FJSSP-W instances, respectively. In Sec. 1, a Mixed Integer Linear Programming (MILP) representation is presented that allows for the application of exact Mathematical Programming solvers like `Gurobi`. Further, an implementation as Constraint Programming (CP) model is provided in Sec. 2. Toolboxes (e.g. `CPLEX`) for this type of problem representation are currently among the most sophisticated solving techniques for Production Scheduling tasks. Finally, a convenient problem representation for Simulation-Based Optimization (SBO) algorithms (e.g. metaheuristics, like `GA`) is presented in Sec 3. It includes the definition of a precise schedule encoding and the design of the production system as an functional simulation environment.

1. Mixed-Integer Linear Programming Formulation

The Mixed Integer Linear Program (MILP) formulation is a common model to approach the FJSSP(-W) problem class using mathematical optimization methods, i.e. branch-and-bound or cutting-plane methods. In this section, the MILP representation is systematically formulated describing the decision variables, constraints and objective functions in detail. All necessary parameters are first specified for the FJSSP before an extension for the FJSSP-W is provided.

1.1. Decision Variables of the FJSSP

In a first step, the decision variables can be defined in the following way:

Notation	Description of decision variables	Domain
$X_{i,j,i',j'}$	Indication of whether operation $O_{i,j}$ is processed **after** operation $O_{i',j'}$ on the same machine: 1 if `True`, 0 else.	$\mathbb{Z}_2$
$Y_{i,j,k}$	The variable indicates whether operation $O_{i,j}$ is processed on machine M_k: 1 if `True`, 0 else.	$\mathbb{Z}_2$
$C_{i,j}$	A variable representing the completion time of the jth operation $O_{i,j}$ in the ith job J_i	$\mathbb{R}_+$

In this context, the notation $\mathbb{Z}_2 = \{0, 1\}$ represents the ring of integers modulo 2 indicating the binary character of a decision variable. Continuous variables, on the other hand, take on positive real numbered values in $\mathbb{R}_+$. As a result, each schedule $\mathbf{y} \in \mathcal{Y}$ is represented by a feasible value assignment of these $(2 + \mathtt{N})\mathtt{N}$ decision variables.

In addition, an index set $M_{i,j}$ is introduced that contains the indices k of the permissible machines M_k $(k = 1, \ldots, m)$ for all operations $O_{i,j}$ $(j = 1, \ldots, n_i)$ of all jobs J_i $(i = 1, \ldots, n)$:

$$M_{i,j} = \{k : O_{i,j} \text{ can be performed on machine } M_k\} \tag{1}$$

Next, the constraints of the FJSSP are described on the basis of these decision variables. To do so, the assignment, sequencing, and variable type constraints are considered independently. Afterwards, we address a typical optimization objective associated constraint formulations.

1.2. General Restrictions in FJSSP

Assignment constraints. It is imperative that the total number of $\mathtt{N} = \sum_{i=1}^{n} n_i$ operations $O_{i,j}$ of an FJSSP instance be executed on exactly one of the machines available and admissible for the respective operation. These are specified by the indices in the set $M_{i,j}$.

$$\sum\nolimits_{k \in M_{i,j}} Y_{i,j,k} = 1 \quad \text{for } 1 \leq i \leq n \text{ and } 1 \leq j \leq n_i \tag{2}$$

By requiring that the sum of $Y_{i,j,k}$ over all admissible machine indices equals 1, this set of $\mathtt{N}$ constraints guarantees that all operations $O_{i,j}$ are assigned to exactly one machine.

Sequencing constraints on the job level. The FJSSP demands that all operations of a single job need to be executed in a predefined sequence. For this type of restrictions, a formulation is proposed that involves the completion times $C_{i,j}$ of the operations within the FJSSP instance. Hence, one can require that the completion time $C_{i,1}$ of the first operation $O_{i,1}$ of job i cannot be smaller than the corresponding processing time (see Eq. (3)). Also, the completion time of the jth operation $O_{i,j}$ of job i cannot be smaller than the sum of the completion time of its predecessor operation $O_{i,j-1}$ in the same job i and its own processing time (cf. Eq. (4)). Consequently, one receives another two sets of $\mathtt{N}$ constraints of the form

$$C_{i,1} \geq \sum\nolimits_{k \in M_{i,1}} Y_{i,1,k} T_{i,1,k} \quad \text{for } 1 \leq i \leq n \tag{3}$$

$$C_{i,j} \geq C_{i,j-1} + \sum\nolimits_{k \in M_{i,j}} Y_{i,j,k} T_{i,j,k} \quad \text{for } 1 \leq i \leq n \text{ and } 2 \leq j \leq n_i. \tag{4}$$

Sequencing constraints on machine level. Taking into account the order of operations on the machine level, it must be ensured that the completion times of two successor operations on the one and same machine are not unacceptably close to each other. For the proper formulation, an auxiliary large penalty parameter L is introduced. This can, for example, be achieved by aggregating the maximal processing times of all operations over all machines

$$L = \sum_{i=1}^{n} \sum_{j=1}^{n_i} \max_{k=1,\ldots,m}(T_{i,j,k}). \tag{5}$$

Accordingly, the following two sets of constraints are obtained

$$C_{i,j} \geq \quad C_{i',j'} + T_{i,j,k} - L \cdot (3 - X_{i,j,i',j'} - Y_{i,j,k} - Y_{i',j',k}) \tag{6}$$

$$C_{i',j'} \geq \quad C_{i,j} + T_{i',j',k} - L \cdot (X_{i,j,i',j'} + 2 - Y_{i,j,k} - Y_{i',j',k}). \tag{7}$$

Both sets (6) and (7) are required to hold for operations $O_{i,j}$ with $1 \leq i < n$ and $1 \leq j \leq n_i$ as well as operations $O_{i',j'}$ with $i' > i$ and $1 \leq j' \leq n_{i'}$ on those machines M_k that satisfy $k \in M_{i,j} \cap M_{i',j'}$.

The two sets of constraints (6) and (7) are trivially fulfilled as long as the bracketed factor multiplied by L is nonzero. In case that operation $O_{i,j}$ and $O_{i',j'}$ are processed on the same machine M_k, i.e. $Y_{i,j,k} = 1$ and $Y_{i',j',k} = 1$, the sequencing between these two specific operations is controlled by the decision variable $X_{i,j,i',j'}$. If $X_{j,k,j',k'} = 1$ holds, operation $O_{i,j}$ is executed after operation $O_{i',j'}$ on the same machine M_k. This implies that the first set of constraints defined by Eq. (6) is active. It ensures that the processing time of the successor operation $O_{i,j}$ does not start before the predecessor operation $O_{i',j'}$ is finished. Conversely, if $O_{i',j'}$ is executed after operation $O_{i,j}$ on the same machine, the decision variable is $X_{i,j,i',j'} = 0$. That is, the second set of constraints (7) is active and guarantees non overlapping processing times of both operations, respectively.

Variable type constraints. This type of constraints restrict the assignment of decision variables to permissible values. They are specified by the definition of the decision variables above. The binary variables $X_{i,j,i',j'}$ and $X_{i,j,k}$ only accept the values 0 and 1 in $\mathbb{Z}_2$. The range of the continuous variables $C_{i,j}$ is restricted to positive real values in $\mathbb{R}_+$. In summary

$$X_{i,j,i',j'} \in \{0,1\}, \qquad Y_{i,j,k} \in \{0,1\}, \quad \text{and} \quad C_{i,j} > 0. \tag{8}$$

Without considering a specific objective function C and potentially associated additional constraints, the restrictions of the FJSSP environment are now adequately described by the constraints in Eqs. (2), (3), (4), (6), (7), and (8).

1.3. Objective function and associated constraints for the FJSSP

Makespan minimization. The total time required for the execution of all jobs in an FJSSP instance is referred to as the makespan of the FJSSP. Its minimization represents one of the

most common objectives considered in this problem domain. The formal formulation of the makespan depends on the representation of the FJSSP and in this specific case coincides with the maximum value C_{max} of the completion times $C_{i,j}$ of all operations $O_{i,j}$. That is, the objective becomes

$$\min\left(C_{\max}\right) = \min\left(\max_{\substack{1 \leq i \leq n \\ 1 \leq j \leq n_i}} C_{i,j}\right) \tag{9}$$

When using the above definition of the makespan, it must be guaranteed that the completion time of the last operation O_{i,n_i} in each job J_i is not exceeding the makespan. Consequently, one gets an additional set of constraints `N` associated with this specific objective

$$C_{max} \geq C_{i,n_i} \qquad \text{for } 1 \leq i \leq n. \tag{10}$$

1.4. Extended MILP Formulation for the FJSSP-W

An obvious generalization of the MILP formulation for the FJSSP is used. The worker flexibility is modeled by an additional set of constraints that follows the same ideas as the sequencing constraints on machine level. The following description explains in detail only the additional notations and constraints of the MILP formulation for the FJSSP-W.

1.4.1. Indices and Variables

To denote the workers the subscripts s, s' with $1 \leq s, s' \leq w$ and w the number of workers are used. The processing time of operation $O_{i,j}$ on machine M_k also depends on worker W_s and is denoted by $T_{i,j,k,s}$

The decision variables $Y_{i,j,k}$ have to be changed to $Y_{i,j,k,s}$ which are 1, if operation $O_{i,j}$ is processed on machine M_k by worker W_s and 0 else. Furthermore an additional type of decision variables is necessary: $U_{i,j,i',j'}$, which is set to 1 if Operation $O_{i,j}$ is processed after operation $O_{i',j'}$ by the same worker. To describe the admissible workers for processing operation $O_{i,j}$ on machine M_k the index set

$$W_{i,j,k} = \{s : \text{ such that } O_{i,j} \text{ can be processed on } M_k \text{ by worker } W_s\}$$

is used.

1.4.2. Constraints

Assignment constraints. These constraints guarantee that all operations $O_{i,j}$ are assigned to exactly one machine and one worker.

$$\sum\nolimits_{k \in M_{i,j}} \sum\nolimits_{s \in W_{i,j,k}} Y_{i,j,k,s} = 1 \text{ for } 1 \leq i \leq n, 1 \leq j \leq n_i$$

Sequencing constraints on job level.

$$C_{i,1} \geq \sum\nolimits_{k \in M_{i,1}} \sum\nolimits_{s \in W_{i,1,k}} Y_{i,1,k,s} T_{i,1,k,s} \text{ for } 1 \leq i \leq n$$

$$C_{i,j} \geq C_{i,j-1} + \sum\nolimits_{k \in M_{i,j}} \sum\nolimits_{s \in W_{i,j,k}} Y_{i,j,k,s} T_{i,j,k,s} \text{ for } 1 \leq i \leq n, 1 < j \leq n_i$$

Sequencing constraints on machine level.

$$C_{i,j} \geq C_{i',j'} + T_{i,j,k,s} - L \cdot (3 - X_{i,j,i',j'} - Y_{i,j,k,s} - Y_{i',j',k,s'})$$

$$C_{i',j'} \geq C_{i,j} + T_{i',j',k,s'} - L \cdot (X_{i,j,i',j'} + 2 - Y_{i,j,k,s} - Y_{i',j',k,s'})$$

Both constraints have to hold for operations $O_{i,j}$ with $1 \leq i < n$ and $1 \leq j \leq n_i$ and operation $O_{i',j'}$ with $i' > i$ and $1 \leq j' \leq n_{i'}$, for machines M_k with $k \in M_{i,j} \cap M_{i',j'}$ and for workers W_s ($s \in W_{i,j,k}$) and $W_{s'}$ ($s' \in W_{i',j',k}$).

Sequencing constraints on worker level.

$$C_{i,j} \geq C_{i',j'} + T_{i,j,k,s} - L \cdot (3 - U_{i,j,i',j'} - Y_{i,j,k,s} - Y_{i',j',k',s})$$

$$C_{i',j'} \geq C_{i,j} + T_{i',j',k',s} - L \cdot (U_{i,j,i',j'} + 2 - Y_{i,j,k,s} - Y_{i',j',k',s})$$

These two sets of constraints have to hold for operations $O_{i,j}$ with $1 \leq i < n$ and $1 \leq j \leq n_i$ and operation $O_{i',j'}$ with $i' > i$ and $1 \leq j' \leq n_{i'}$, for machines M_k ($k \in M_{i,j}$) and $M_{k'}$ ($k' \in M_{i',j'}$) and for workers W_s with $s \in W_{i,j,k} \cap W_{i',j',k'}$. For the large number L, one can use

$$L = \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{n_i} \max\nolimits_{1 \leq k \leq m; 1 \leq s \leq w} (T_{i,j,k,s})$$

The idea of these constraints is the same as in Eq. (6) and (7) and explained there in detail. Finally makespan and variable type constraint are the same as in 1.3.

2. Constraint Programming Formulation

The FJSSP can be easily formulated as a discrete single-objective Constraint Optimization Problem (COP). The two Constraint Programming (CP) solvers that we take into account in this comparison implement variable types, like interval variables and constraints like no-overlap or end-before-start, that are tailored to deal with scheduling problems of general form.

The investigations included the solvers CP-SAT, i.e. an open-source solver developed by Google, and CP-Optimizer (CPO), a constraint optimization solver provided as a part of the IBM ILOG CPLEX Optimization Studio. The problem formulation for both solvers uses the same subscripts, variables, and index sets as the MILP formulation (see 1).

2.1. Variables

The data is given by $T_{i,j,k}$ the processing time of operation $O_{i,j}$ on machine k. The variables are an interval variable $I(i,j)$ with start-time $start(i,j)$, end-time $end(i,j)$ and duration $duration(i,j)$ and an alternative interval variable $a\_I(i,j,k)$ for $k \in M_{i,j}$ to represent feasible processing modes, i.e. the selectable machines. For these intervals $a\_duration(i,j,k) = T_{i,j,k}$ holds. The CP-SAT solver additionally uses a boolean variable $a\_presence(i,j,k)$ which indicates whether the alternative interval $a\_I(i,j,k)$ is chosen.

2.2. Objective

The objektive, i.e. minimizing the makespan, is formulated as

$$\max_{1 \leq i \leq n} (end\_of(I_{i,n_i})) \rightarrow \min$$

2.3. Constraints

Assignment constraints. The assignment of an operation $O_{i,j}$ to a machine is done by linking the interval variable $I(i,j)$ to an alternative interval variable $a\_I(i,j,k)$.

This can be done by using the *alternative* constraint of CPO. For $1 \leq i \leq n$ and $1 \leq j \leq n_i$ we add

$$alternative\,(I(i,j), [a\_I(i,j,k) : \text{ for } k \in M_{i,j}])\,.$$

For the CP-SAT solver the *exactly_one* constraint is used. For $1 \leq i \leq n$ and $1 \leq j \leq n_i$ we add

$$\begin{aligned} exactly\_one\,(a\_presence(i,j,k) : \text{ for } k \in M_{i,j}) \\ start(i,j) = a\_start(i,j,k) \text{ if } a\_presence(i,j,k) = True \\ duration(i,j) = a\_duration(i,j,k) \text{ if } a\_presence(i,j,k) = True \\ end(i,j) = a\_end(i,j,k) \text{ if } a\_presence(i,j,k) = True \end{aligned}$$

Sequencing constraints on job level. The sequencing constraints within jobs are formulated slightly different for the two solvers.
The CPO solver provides an end_before_start constraint,

$$end\_before\_start(I(i,j-1), I(i,j)) \text{ for } 1 \leq i \leq n \text{ and } 2 \leq j \leq n_i$$

and with the CP-SAT solver, start and end times are used

$$start(i,j) \geq end(i,j-1) \text{ for } 1 \leq i \leq n \text{ and } 2 \leq j \leq n_i$$

Sequencing constraints on machine level. The *no_overlap* constraint which is provided by both solvers makes it very easy to formulate the sequencing constraint within machines. For $1 \leq k \leq m$ the following constraints are added:

$$no\_overlap\,(a\_I(i,j,k) : \text{ for } 1 \leq i \leq n, 1 \leq j \leq n_i \text{ if } k \in M_{i,j})$$

2.4. Extending the CP Formulation for the FJSSP-W

The same as for the MILP formulation holds also for the CP formulation: The FJSSP-W is an obvious generalization of the FJSSP. In the following the formulation for CPO solver is presented. The necessary changes for using the CP-SAT solver are obvious when the two different formulations for the FJSSP are kept in mind. Again, each operation is represented by an interval variable $I(i,j)$ with start-time $start(i,j)$, end-time $end(i,j)$ and duration $duration(i,j)$. For the FJSSP-W an alternative interval variable $a\_I(i,j,k,s)$ has to be defined for each admissible machine-worker combination.

The assignment of an operation $O_{j,k}$ to exactly one machine-worker combination is done by linking the interval variable $I(j,k)$ to an alternative interval variable $a\_I(i,j,k,s)$. This can be done by the *alternative* constraints

$$alternative\,(I(i,j), [a\_I(i,j,k,s) : \text{ for } k \in M_{i,j} \text{ and } s \in W_{i,j,k}])$$

for $1 \leq i \leq n$ and $1 \leq j \leq n_i$.

The sequencing constraints on job level are:

$$end\_before\_start(I(i,j-1), I(i,j)) \text{ for } 1 \leq i \leq n \text{ and } 2 \leq j \leq n_i$$

and to guarantee an admissible sequencing at the machine level, the following constraints are added for $1 \leq k \leq m$:

$$no\_overlap\,(a\_I(i,j,k,s) : \text{ for } 1 \leq i \leq n, 1 \leq j \leq n_i \text{ if } k \in M_{i,j} \text{ and } s \in W_{i,j,k})$$

Additionally the sequencing on worker level has to be guaranteed by using another set of *no_overlap* constraints. For $1 \leq s \leq w$:

$$no\_overlap\,(a\_I(i,j,k,s) : \text{ for } 1 \leq i \leq n, 1 \leq j \leq n_i, k \in M_{i,j} \text{ if } s \in W_{i,j,k})$$

3. Simulation-Based Optimization Attempt

Multiple problem encodings can be found in the literature, the most common one being [1, 2]. To encode the information using this system, each schedule $\mathbf{y} = (\mathbf{s}|\mathbf{a}) \in \mathcal{Y} \subset \mathbb{N}^{N+N}$ is encoded by two distinct vectors $\mathbf{s}$, and $\mathbf{a}$. The sequence vector $\mathbf{s} \in \mathbb{N}^{N}$ refers to the job sequence of the schedule and determines the order of the operations on their assigned machines. A feasible $\mathbf{s}$ contains each job index i exactly n_i times and the jth appearance of index i corresponds to operation O_{ij}. Since the sequence of operations of a job is fixed, the job dependent sequence will always be feasible. The assignment vector $\mathbf{a} \in \mathbb{N}^{N}$ contains the corresponding machine assignments. The order in $\mathbf{a}$ is fixed in the beginning and never changes to make sure each element of $\mathbf{a}$ can only be assigned with available machines for every operation. This, together with the encoding chosen for $\mathbf{s}$ ensures that all solutions produced either randomly or through recombination and mutation will always be feasible. For example, for a problem with 2 jobs, job 1 consisting of 3 operations and job 2 consisting of 2 operations, the sequence vector must contain the values $\{\mathbf{1}, \mathbf{1}, \mathbf{1}, \mathbf{2}, \mathbf{2}\}$ in any order. Assuming there are 2 machines in the production environment and the instance is fully flexible (every operation can be performed on any machine), the assignment vector can contain any combination of 1 and 2, with a length corresponding to the amount of operations. The order of the values is fixed, starting from the first operation of the first job, moving to the last operation to the first job, then moving on to the first operation of the second job and moving through to the end to the last operation of the second job. One example for the described problem could look like Listing 1.

```
s = [1, 2, 1, 1, 2]
a = [2, 2, 1, 2, 1]
```

Listing 1: Example sequence and assignment vector

Assuming a processing time of 2 for every job, this results in the schedule depicted in Fig. 1. To evaluate the solution, the schedule is constructed according to the instructions

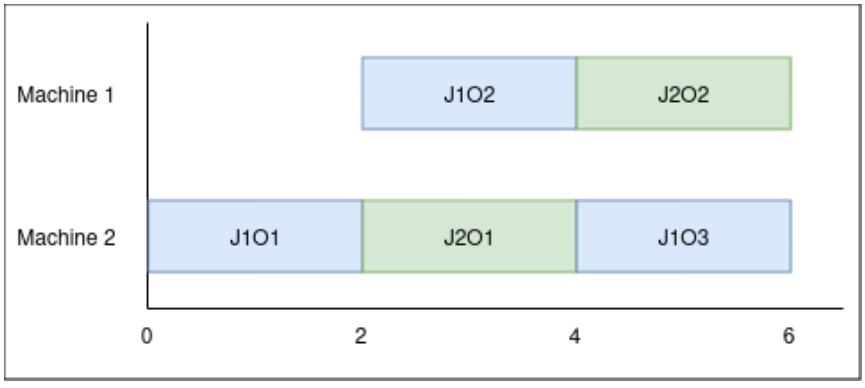

Figure 1: Example schedule produced by Listing 1.

in **s** and **a**. To construct the schedule, the operations are inserted according to the sequence in **s**. The start time t_{ij} is determined based on the current last operation ending time on the assigned machine $t_{a_{ij}}$ and the last ending time in the job sequence of the operation t_{prev} (Eq. (11)).

$$t_{prev} = \begin{cases} t_{i,j-1} + d_{i,j-1} & \text{if } j > 0 \\ 0 & \text{if } j = 0 \end{cases} \tag{11}$$

$t_{i,j}$ is then determined by Eq. (12).

$$t_{i,j} = \max(t_{prev}, t_{a_{i,j}}). \tag{12}$$

$t_{a_{i,j}}$ is replaced (Eq. (13)) and the next operation is inserted into the schedule.

$$t_{a_{i,j}} = t_{i,j} + d_{i,j}. \tag{13}$$

After all starting times have been determined, the fitness of the solution C_{max} is determined by Eq. (14).

$$C_{\max} = \max_{\substack{1 \le i \le n \\ 1 \le j \le n_i}} (t_{i,j} + d_{i,j}) \tag{14}$$

3.1. Extension of SBO Encoding for FJSSP-W

The encoding for simulation-based approaches stays the same for sequence and machine assignments as for the FJSSP problem. To add the worker flexibility, an additional vector $\mathbf{w} \in \mathbb{N}^{\mathbb{N}}$ is added. The vector has the same properties as the machine assignment vector **a**, in that it is in the same fixed order and contains the IDs of the workers assigned to the operation-machine combination chosen to do the task. Extending the example used in Listing 1 with 3 available workers for the system, a problem instance could look like Listing 2.

```
s = [1, 2, 1, 1, 2]
a = [2, 2, 1, 2, 1]
w = [3, 1, 1, 2, 3]
```

Listing 2: Example sequence, assignment and worker vector

The resulting schedule looks as depicted in Fig. 2.

References

[1] H. E. Nouri, O. Belkahla Driss, K. Ghédira, Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model, Journal of Industrial Engineering International 14 (1) (2018) 1–14. doi:10.1007/s40092-017-0204-z.
URL `http://link.springer.com/10.1007/s40092-017-0204-z`

[2] D. Hutter, T. Steinberger, M. Hellwig, An Interior-point Genetic Algorithm with Restarts for Flexible Job Shop Scheduling Problems, in: 2024 IEEE Congress on Evolutionary Computation (CEC), 2024, pp. 01–09. doi:10.1109/CEC60901.2024.10611934.

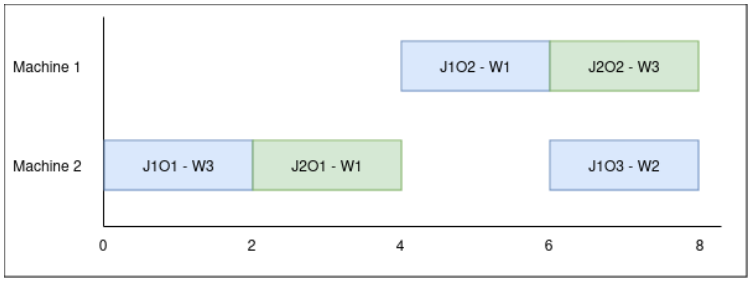

Figure 2: Example schedule produced by Listing 1.