Flexible Distributed Job Shop Scheduling Problem (FDJSSP)

Feodor Pisnitchenko

July 15, 2025

1 Objective

The primary objective of the FDJSSP is to **minimize the makespan**, i.e., the completion time of the last operation in the distributed job shop. The makespan is the time by which all operations have finished. Since FDJSSP generalizes the classical Job Shop Scheduling Problem, it is NP-hard and becomes intractable for large instances. Hence, participants should develop **efficient heuristic or metaheuristic algorithms** that produce high-quality schedules within reasonable computational time. These algorithms must optimize the assignment of operations to machines, the sequencing on each machine, and the scheduling of any required data transfers, with the goal of minimizing the makespan.

This problem is practically relevant in domains like distributed computing (e.g., scheduling tasks across GPU clusters) and manufacturing (e.g., multi-site production), where tasks depend on each other and data may need to be communicated between different locations.

2 Background

The FDJSSP is a significant extension of the classical Job Shop Scheduling Problem (JSSP), a cornerstone of operations research. It enhances the traditional model by incorporating two layers of complexity that reflect modern operational realities.

- **Flexible Machine Assignment:** Each operation can be processed on one of several candidate machines. This flexibility can improve resource utilization but makes the assignment decision more complex.
- **Distributed Environment:** Machines are distributed across a network of sites. Dependent operations assigned to different machines incur communication delays when transferring data. For example, in distributed deep learning, different layers of a neural network may run on different GPUs, and intermediate results (gradients, activations) must be sent over a network.

These extensions introduce additional scheduling considerations. In particular:

- **Operation Dependencies:** The operations form a Directed Acyclic Graph (DAG) of precedence constraints. If $(i_1, i_2) \in B$, then operation i_2 cannot start until operation i_1 finishes and its output is available.
- **Communication Delays:** If two operations with a precedence relation are assigned to different machines, the data transfer between those machines takes time. Each data transfer has a duration (based on data size and channel bandwidth) and must be explicitly scheduled. All communication channels have the same bandwidth, and all machines are identical.

- **Scheduling Constraints:** Any valid schedule must respect:
  - **Precedence:** An operation can only start after all its predecessors have finished (including any required data transfers).
  - **Machine Exclusivity:** No two operations may run at the same time on the same machine.
  - **Channel Exclusivity:** No two data transfers may use the same communication link simultaneously.

These considerations make FDJSSP analogous to scheduling problems in distributed training or multi-factory operations, where coordinating tasks and data movement is crucial for performance.

3 Mathematical Model

We present a mixed-integer programming (MIP) formulation of FDJSSP. The notation is as follows:

Sets:

- I: set of operations, $|I| = n$.
- J: set of machines, $|J| = m$.
- $A_i \subseteq J$: set of candidate machines for operation $i \in I$.
- $B \subseteq I \times I$: set of precedence edges; $(i_1, i_2) \in B$ means i_1 must finish before i_2 starts.
- $C = \{(j_1, j_2) \in J \times J \mid j_1 \neq j_2\}$: set of directed communication channels. We assume the network is fully connected without self-loops.

Parameters:

- D_i^o: processing duration of operation i.
- D_{i_1,i_2}^c: communication duration (data transfer time) required if $(i_1, i_2) \in B$ and the operations are on different machines. If i_1 and i_2 are on the same machine, no communication is needed.
- M: a sufficiently large constant (big-M) used to model disjunctive constraints.

Decision Variables:

- $s_i, e_i \geq 0$: start time and end time of operation i.
- $x_{i,j} \in \{0, 1\}$: 1 if operation i is assigned to machine j, 0 otherwise.
- $c_{i_1,i_2}, d_{i_1,i_2} \geq 0$: start time and end time of the communication (data transfer) for the precedence $(i_1, i_2) \in B$. (If i_1 and i_2 are on the same machine, we set $c_{i_1,i_2} = d_{i_1,i_2} = e_{i_1}$.)
- $z_{i_1,i_2,j_1,j_2} \in \{0, 1\}$: 1 if the transfer for $(i_1, i_2) \in B$ uses channel $(j_1, j_2) \in C$, 0 otherwise.
- $y_{i_1,i_2} \in \{0, 1\}$: 1 if operation i_1 precedes i_2 on their assigned machine, 0 otherwise.
- $w_{i_1,i_2,i_3,i_4} \in \{0, 1\}$: 1 if the communication for $(i_1, i_2) \in B$ precedes that for $(i_3, i_4) \in B$ on the same channel, 0 otherwise.

- $z \geq 0$: makespan.

The MIP model:

Objective: Minimize makespan z.

$$\min z$$

Subject to:

1. **Makespan:**

$$z \geq e_i \quad \forall i \in I.$$

Ensures z is at least the completion time of every operation.

2. **Operation Duration:**

$$e_i = s_i + D_i^o \quad \forall i \in I.$$

Sets the end time of each operation equal to its start time plus processing duration.

3. **Machine Assignment:**

$$\sum_{j \in A_i} x_{i,j} = 1 \quad \forall i \in I.$$

Each operation must be assigned to exactly one candidate machine.

4. **Machine Scheduling (Disjunctive Constraints):**

For any two distinct operations $i_1 \neq i_2$ and any machine $j \in J$:

$$e_{i_1} \leq s_{i_2} + M\big(3 - y_{i_1,i_2} - x_{i_1,j} - x_{i_2,j}\big)$$

A symmetric constraint handles the opposite ordering:

$$e_{i_2} \leq s_{i_1} + M\big(3 - y_{i_2,i_1} - x_{i_1,j} - x_{i_2,j}\big)$$

This enforces that if $x_{i_1,j} = x_{i_2,j} = 1$ (both operations on machine j) and $y_{i_1,i_2} = 1$, then $e_{i_1} \leq s_{i_2}$.

5. **Operation Order on Machines:**

$y_{i_1,i_2} + y_{i_2,i_1} = 1 \quad \forall i_1 \neq i_2.$

This ensures that any two operations have a strict order if they are on the same machine (the big-M makes it loose otherwise).

6. **Channel Selection (Linearization):**

For each $(i_1, i_2) \in B$ and $(j_1, j_2) \in C$:

$$z_{i_1,i_2,j_1,j_2} \leq x_{i_1,j_1}, \quad z_{i_1,i_2,j_1,j_2} \leq x_{i_2,j_2}, \quad z_{i_1,i_2,j_1,j_2} \geq x_{i_1,j_1} + x_{i_2,j_2} - 1.$$

This ensures $z_{i_1,i_2,j_1,j_2} = x_{i_1,j_1} x_{i_2,j_2}$ if $j_1 \neq j_2$; automatically, $\sum_{(j_1,j_2) \in C} z_{i_1,i_2,j_1,j_2} = 1$ if different machines, 0 if same.

7. **Communication Duration:**

$$d_{i_1,i_2} = c_{i_1,i_2} + D^c_{i_1,i_2} \cdot \sum_{(j_1,j_2)\in C} z_{i_1,i_2,j_1,j_2} \quad \forall (i_1,i_2) \in B.$$

Ensures the end time of each transfer is its start time plus the required transfer duration if on different machines, or equal to the start time if same.

8. **Start After Transfer:**

$$s_{i_2} \geq d_{i_1,i_2} \quad \forall (i_1,i_2) \in B.$$

An operation cannot start until its incoming data transfer is completed.

9. **Transfer After Operation:**

$$c_{i_1,i_2} \geq e_{i_1} \quad \forall (i_1,i_2) \in B.$$

The data transfer cannot start until the sending operation finishes.

10. **Channel Scheduling (Data Transfers):**

For any two distinct edges $(i_1,i_2) \neq (i_3,i_4) \in B$ and any channel $(j_1,j_2) \in C$:

$$c_{i_3,i_4} \geq d_{i_1,i_2} - M\big(3 - w_{i_1,i_2,i_3,i_4} - z_{i_1,i_2,j_1,j_2} - z_{i_3,i_4,j_1,j_2}\big).$$

Combined with the symmetric constraint (swapping (i_1,i_2) and (i_3,i_4)), this ensures that if both transfers use the same channel (j_1,j_2), they do not overlap.

11. **Order on Channels:**

$$w_{i_1,i_2,i_3,i_4} + w_{i_3,i_4,i_1,i_2} = 1 \quad \forall (i_1,i_2) \neq (i_3,i_4).$$

Ensures a strict order for any two transfers if they share a channel; the big-M makes it loose otherwise.

12. **Domains:**

$$s_i, e_i, c_{i_1,i_2}, d_{i_1,i_2}, z \geq 0;$$
$$x_{i,j}, z_{i_1,i_2,j_1,j_2}, y_{i_1,i_2}, w_{i_1,i_2,i_3,i_4} \in \{0,1\}.$$

The objective and constraints (1)-(3) define makespan, operation timings, and assignments; (4)-(5) ensure non-overlap on machines; (6)-(9) model the creation and timing of data transfers; and (10)-(11) enforce non-overlap of communications on each channel.

4 Tasks

Literature Review: Conduct a survey of recent (2020–present) research on flexible or distributed job shop scheduling, emphasizing heuristic or metaheuristic methods. Identify works that handle flexible machine assignment and/or communication or transportation delays. Summarize the techniques and their relevance to FDJSSP.

2. **Heuristic Algorithm Design:** Propose a heuristic or metaheuristic algorithm to solve FDJSSP. Describe how your approach handles: (a) assigning operations to machines, (b) sequencing operations on each machine, and (c) scheduling necessary data transfers (channels). Include pseudocode or a flowchart of the algorithm.

3. **Implementation and Testing:** Implement your algorithm. Use the provided instance generator to create test cases of varying sizes (different numbers of operations and machines). Apply your algorithm to these instances.

4. **Performance Evaluation:** Compare your algorithm against baseline methods (such as a simple greedy scheduler or an exact MILP solver on small instances). Check the makespan and runtime for each instance. Evaluate how solution quality and runtime scale as the problem size grows.

5 Literature Review

- **Dauzère-Pérès et al. (2024)** – "*The flexible job shop scheduling problem: A review.*" European Journal of Operational Research, 314(2), 409–432.

  Review of flexible JSSP, highlighting heuristic methods like genetic algorithms and local search.

- **Li et al. (2020)** – *"An improved Jaya algorithm for solving the flexible job shop scheduling problem with transportation and setup times."* Knowledge-Based Systems, 200.

  Extends FJSP to include transportation delays between machines.

- **Ren et al. (2021)** – *"Joint optimization for dynamic flexible job-shop scheduling problem with transportation time and resource constraints."* International Journal of Production Research, 60(18).

  Dynamic FJSP with transportation times.

- **Luo et al. (2022)** – *"A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm."* Expert Systems with Applications, 207:117984.

  Distributed FJSP with worker considerations.

- **Yuan et al. (2025)** – *"Distributed heterogeneous flexible job-shop scheduling problem considering automated guided vehicle transportation via improved deep Q network."* Swarm and Evolutionary Computation, 94:101902.

  Applies reinforcement learning (deep Q-network) to a distributed flexible JSSP with automated guided vehicle (AGV) transport.