PyJobShop: Solving machine scheduling problems with constraint programming in Python

Leon Lan   Joost Berkhout

Vrije Universiteit Amsterdam, The Netherlands

April 5, 2024

As a fresh and young PhD student in 2021...

3-year industry project: animal feed manufacturing

...I faced many challenges

- Industrial scheduling problem with many constraints
- MILP models are too slow
- Lack of open-source research code

Recent developments

- Constraint programming (CP) outperforms MILP (Naderi et al., 2023)
- Built a CP prototype for our industry partner
- Development of an open-source vehicle routing solver PyVRP (Wouda et al., 2024)

PyJobShop: A Python library for modeling and solving real-world machine scheduling problems

- Simple modeling interface for scheduling (model-and-run)
- CP Optimizer as underlying solver (Laborie et al., 2018)
- Open-source, unit-tested and extensible

PyJobShop's model is based on the flexible job shop problem

The flexible job shop problem is defined by the following objects:

- **Job**: collection of operations used to measure performance
- **Operation**: actual tasks to be scheduled
- **Machine**: resources that can process operations

Decisions to make:

- Operation: machine assignment, start time, completion time
- Machine: sequencing of assigned operations

Goal: minimize an objective function and satisfy all constraints

(See survey by Dauzère-Pérès et al., 2024.)

Job, operation and machine attributes

Job

- Weights
- Release dates
- Due dates
- Deadlines

Operation

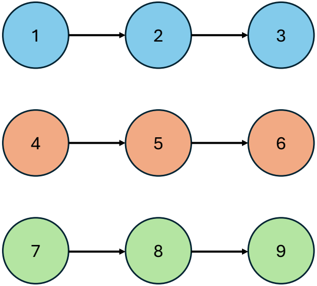
- Eligible machines
- Processing times
- Earliest/latest start time
- Earliest/latest end time

Machine

- Downtimes
- Setup times
- ...
- ...

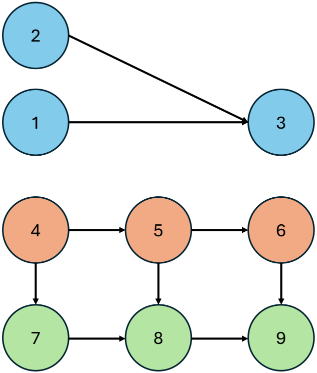
The *operations graph* defines all pairwise relationships between operations

Classic FJSP (linear routing)



The *operations graph* defines all pairwise relationships between operations

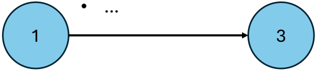
Arbitrary precedence graph (sequencing flexibility)



The *operations graph* defines all pairwise relationships between operations

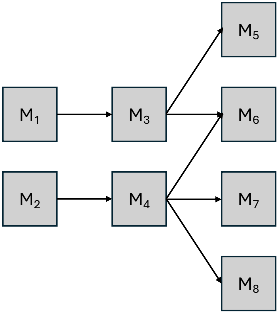
Constraint types

- End before start
- End before end
- End at start
- End at end
- ...



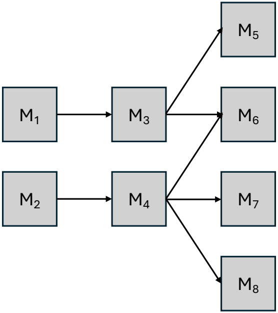
- Same machine
- Different machine
- Accessible machine

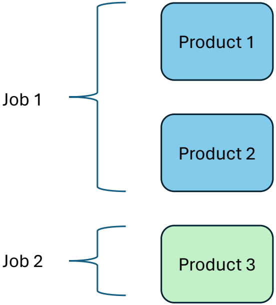
The *machines graph* defines pairwise relationships between machines



Example hybrid flow shop from the animal feed industry

Input data




Job 1

Product 1

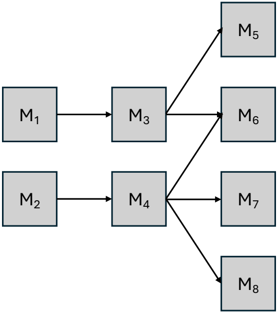
Product 2

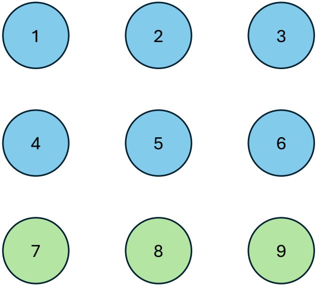
Job 2

Product 3

Example hybrid flow shop from the animal feed industry

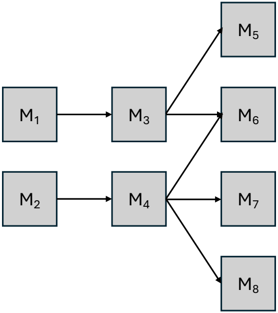
Operations

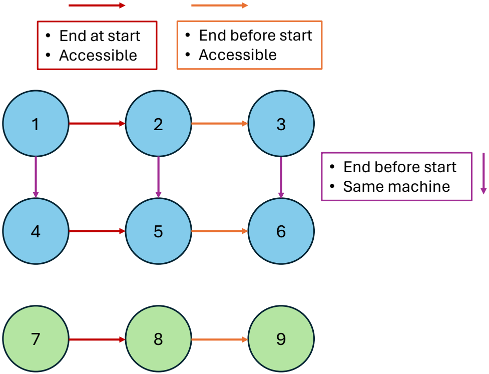




Example hybrid flow shop from the animal feed industry

Operations graph




- End at start
- Accessible

- End before start
- Accessible

- End before start
- Same machine

Solving FJSP with PyJobShop's modeling interface

```python
import random
from pyjobshop import Model, Constraint

m = Model()

jobs = [m.add_job() for _ in range(2)]
machines = [m.add_machine() for _ in range(4)]

for job in jobs:
    operations = [m.add_operation(job=job) for _ in range(4)]

    for machine in machines:
        for op in operations:
            duration = random.randint(1, 10)
            m.add_processing_time(machine, op, duration)

    for idx in range(len(operations) - 1):
        op1, op2 = operations[idx], operations[idx + 1]
        m.add_edge(op1, op2, Constraint.END_BEFORE_START)

result = m.solve()
```

There are many more extensions possible...

- Arbitrary objective functions
- Batching → *b*-batching
- Processing plans (AND/OR graphs) → distributed scheduling
- Resources → RCPSP
- Processing modes → MM-RCPSP
- Google OR-Tools
- Meta- or matheuristic

Thank you!

PyJobShop: Solving machine scheduling problems with constraint programming in Python

Leon Lan   Joost Berkhout

https://github.com/PyJobShop/PyJobShop

l.lan@vu.nl | leonlan | www.leonlan.com