Job-shop scheduling (ft10)

Job-shop scheduling (ft10)

How can a manufacturing company complete the processing of orders (jobs) in minimal time when each order requires the use of specific machines in a specific sequence?

The Job-shop Scheduling Problem is one of the best-known machine scheduling problems. In the classical version of the problem, a finite set of jobs is processed on a finite set of machines. Each job is characterized by a fixed order of operations. Each operation is to be processed on a specific machine for a specified duration. Each machine can process at most one operation at a time. Once an operation initiates processing on a particular machine, the operation must complete processing on that machine uninterrupted.

The objective of the problem is to find a schedule that minimizes the end time (makespan) of the schedule.

This problem is an example of a scheduling problem in which operations are represented by special variables that are called interval variables and interval sequence variables. An interval has a start time, an end time, and a duration. An interval sequence variable (also called a sequence variable) represents a total ordering of the interval variables of the set. There are special constraints for intervals and sequences, including precedence constraints and no overlap constraints, which are used to model this problem.

The data

The instance of job-shop scheduling problem under consideration is the famous `ft10` proposed in 1963[1]. This instance remained unsolved for a quarter century until an optimal solution was proven in 1989[2].

This problem instance consists of 10 jobs with 10 operations per job and 10 machines.

The following table describes each operation of each job. For example, the first job consists of a first operation with a duration of 29 using Machine 0 followed by a second operation with a duration of 78 using Machine 1.

Table 1. Machine number and duration for each operation of each job.

	Op 1	**Op 2**	**Op 3**	**Op 4**	**Op 5**	**Op 6**	**Op 7**	**Op 8**	**Op 9**	**Op 10**
Job 1	0 29	1 78	2 9	3 36	4 49	5 11	6 62	7 56	8 44	9 21
Job 2	0 43	2 90	4 75	9 11	3 69	1 28	6 46	5 46	7 72	8 30
Job 3	1 91	0 85	3 39	2 74	8 90	5 10	7 12	6 89	9 45	4 33
Job 4	1 81	2 95	0 71	4 99	6 9	8 52	7 85	3 98	9 22	5 43
Job 5	2 14	0 6	1 22	5 61	3 26	4 69	8 21	7 49	9 72	6 53
Job 6	2 84	1 2	5 52	3 95	8 48	9 72	0 47	6 65	4 6	7 25

1. H. Fisher and G.L. Thompson. Industrial Scheduling, "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules", pp225-251. Prentice-Hall, 1963.
2. J. Carlier and E. Pinson. "An algorithm for solving the job-shop problem". Management Science, Vol 35, No 2, pp164-176. 1989.

Table 1. Machine number and duration for each operation of each job. (continued)

	Op 1	Op 2	Op 3	Op 4	Op 5	Op 6	Op 7	Op 8	Op 9	Op 10
Job 7	1 46	0 37	3 61	2 13	6 32	5 21	9 32	8 89	7 30	4 55
Job 8	2 31	0 86	1 46	5 74	4 32	6 88	8 19	9 48	7 36	3 79
Job 9	0 76	1 69	3 76	5 51	2 85	9 11	6 40	7 89	4 26	8 74
Job 10	1 85	0 13	2 61	6 7	8 64	9 76	5 47	3 52	4 90	7 45

The model

To model this problem, the unknowns, constraints, and objective must be determined. The unknowns are the times that the operations will start/end. In this model, there are constraints on the order of the operations for each job. There is also a set of constraints to represent that a machine can process only one operation at a time. The objective is to minimize the end (completion) time of the final task.

Modeling the variables

The model uses a set of interval variables to represent the operations. For instance, the first operation of the first job is represented as an interval variable `Op_1_1`. The size of the interval variable represents the duration of the operation:

```
Op_1_1 = intervalVar(size=29);
```

Each machine is represented by a sequence variable that is defined on the set of operations that use the machine. For instance, sequence variable `Machine_0` is defined as:

```
Machine_0 = sequenceVar([Op_1_1, Op_2_1, Op_3_2, Op_4_3, Op_5_2,
                                   Op_6_7, Op_7_2, Op_8_2, Op_9_1, Op_10_2]);
```

Note: The sequence variable alone does not enforce any constraint on the relative position of interval end points. For instance, an interval variable *a* could be sequenced before an interval variable *b* in a sequence *p* without any impact on the relative position between the start/end points of *a* and *b*.

Modeling the constraints

The job-shop scheduling problem involves two types of constraints: precedence constraints between operations and no-overlap constraints on machines.

Precedence constraints between consecutive operations of a job are expressed with `endBeforeStart`. For instance:

```
endBeforeStart(Op_1_1, Op_1_2);
```

states that `Op_1_1` must end before `Op_1_2` starts.

Because a particular machine can perform only one operation at a time, the interval variables that represent the operations that use the machine cannot overlap. This restriction is expressed with `noOverlap` constraints, such as:

```
noOverlap(Machine_0);
```

Modeling the objective function

The objective of this problem is to minimize the end time (makespan) of the schedule. The makespan is defined as the end time of the last operation of the schedule. In other words, the goal is to minimize the maximum of the end times of the intervals that represent the final operations of each of the jobs:

```
minimize(max([endOf(Op_1_10), endOf(Op_2_10), ..., endOf(Op_10_10)]));
```

The solution

For problem `ft10`, the optimal solution has a makespan value of 930.