TabuSearch: Job-shop scheduling problem

Jerzy Wroczyński

2020-06-04

1 Introduction

The job shop scheduling problem is one of many theoretic scheduling problems. In a paper by Dell'Amico and Trubian [2] it was classified as $J||C_{\max}$ using the notation introduced by R.L.Graham et al. [3]. Letter J represents „job shop scheduling problem", two vertical lines with nothing in between mean no further job characteristics are given and $C_{\max}$ defines the optimization problem as minimizing the maximum completion time of all given jobs.

Of course, there are many different types of such problems e.g. there can be a predetermined quantity of machines e.g. only one machine, jobs can have certain characteristics e.g. each job has a *fuzzy due date* etc. but in this paper the problem classified in the previous paragraph will be examined.

We are given following resources:

1. a set J of n jobs to schedule,
2. a set $O = \{1, \ldots, N\}$ of N atomic operations
3. a set M of m machines.

For each job J_j there is a sequence of operations $O_{i,j} \in O$ and each of these operations has to be processed without interruption separately on a machine $\mu_{i,j} \in M$ for $d_{i,j}$ units of time.

For better understanding of a such schedule problem a visual aid of a Gantt chart can be used:

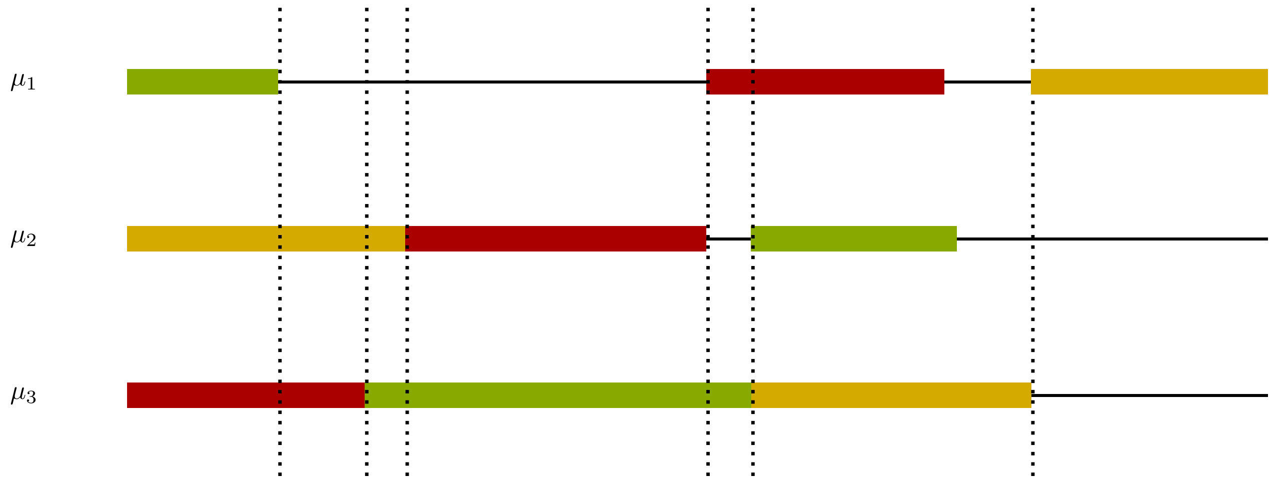

Figure 1: Example Gantt chart

which represents one feasible solution for scheduling three jobs (each consisting of three operations) on three machines. We can see now, there are some difficulties to overcome when dealing with such a problem. For example, in this solution machine μ_1 runs idle for a very long time which can be an indicator on how good this solution is. Our main goal is to minimize the running time of the machine that completes its tasks last.

2 Problem representation

The Gantt chart very clearly shows the amount of time each task takes and for how long each machine runs idle. However, it is not a convenient way of representing this problem when it comes to applying TabuSearch.

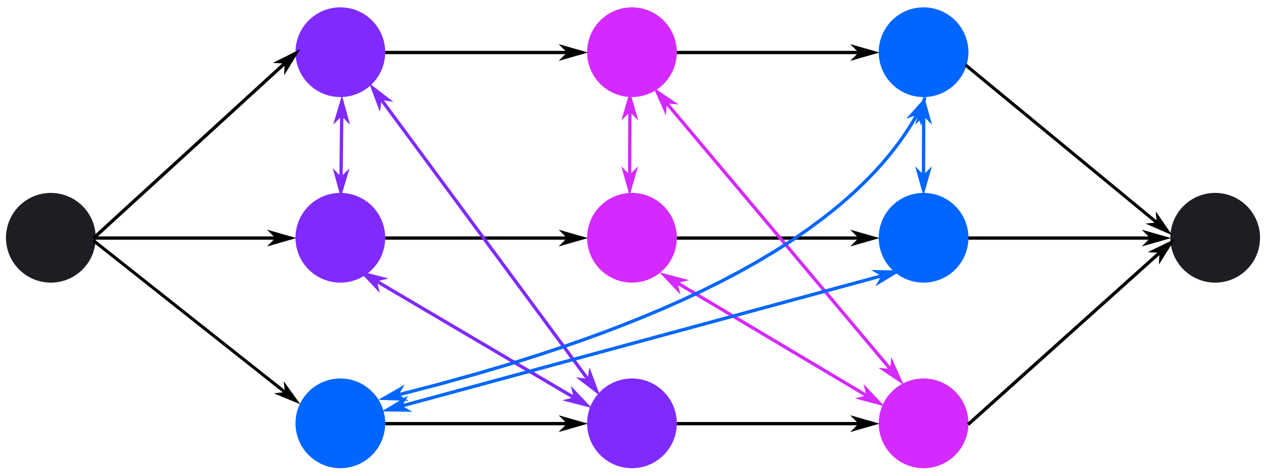

Figure 2: Example disjunctive graph

Figure 2 shows an example of a disjunctive graph that is commonly used when dealing with the job shop problem. Here, each atomic operation is represented as a node. The order of operations in each job is maintained using directed edges (black edges) drawn between appropriate operations (so they form a sequence of operations). These edges are called „conjunctive". The rest of the edges (coloured edges) are called „disjunctive" and they represent all possible combinations of a job schedule. In this abstract form we can think of them as normal, undirected edges. When a schedule is being sought these edges need to become directed as to define order of execution on each of the given machines.

When searching for a solution the original order of operations in a job has to be maintained – direction of black edges cannot be altered. Defining the direction of the undirected edges will give us a certain schedule thus a solution to our problem. One way of directing these edges is to apply an „acyclic orientation" to each group of operations executed on the same machine. Then, having created such a DAG (Directed Acyclic Graph) the longest path (with respect to the cost function) defines the quality of this solution. In other words – the goal here is to minimize the cost of this longest path in our DAG.

3 TabuSearch

The heuristic we will be applying to this problem is TabuSearch. This heuristic algorithm is an extension to LocalSearch which searches through all neighbouring solutions of the currently chosen best solution. It chooses the next solution based on the cost function. TabuSearch introduces so called „Tabu" array that holds all forbidden „moves" that cannot be performed when generating a neighbourhood of the currently chosen solution. This reduces the amount of solutions to consider and thus gives more time for greater expansion in the global solution space.

To prevent from skipping some solutions due to the Tabu array we can add an aspiration criteria. For example if a given move has been put into Tabu array a long time ago, we can again use this move when generating a new neighbourhood.

4 The algorithm

Before running the TabuSearch heuristic we need to define a way of finding the first feasible solution to start from. Using the *Shortest Processing Time* rule from Brandimarte [1, section 2.1] we can generate the first feasible solution.

```
1: //TODO: pick a concrete algorithm
```

References

[1] Paolo Brandimarte. *Routing and scheduling in a flexible job shop by tabu search.* 1993. doi: 10.5555/160231.160247.

[2] Mauro Dell'Amico and Marco Trubian. *Applying tabu search to the job-shop scheduling problem.* Politecnico di Milano, 1-20133 Milano, Italy, 1993. doi: 10.1007/BF02023076.

[3] R.L.Graham, E.L.Lawler, J.K. Lenstra, and A.H.G. Rinnooy Kan. *Optimization and Approximation in Deterministic Sequencing and Scheduling: a Survey.* Elsevier B.V., 1979. doi: 10.1016/S0167-5060(08)70356-X.