ALCOM-FT Deliverable D19
Production Planning Software Prototype ⋆

Peter Lennartz

Institute of Information and Computing Sciences,
Universiteit Utrecht,
The Netherlands

1 Introduction

The *job shop scheduling* problem is the problem of finding a way to schedule a number of operations, such that the last operation is completed as soon as possible. Here an *operation* is a task that must be executed on a resource, called the *machine*.

There are often some relations between the operations, like precedence constraints, which indicate that an operation has to be finished before another can start. For example, consider the case that you were to produce a medical vaccine. First, you would have to mix and process the ingredients of the vaccine before you can go and package it into the right doses.

But there might be another type of constraints, so called no-wait constraints. They indicate that a certain intermediate product has to be processed immediately (or a fixed amount of time later) on the next machine. In our vaccine example the step from the production to the packing could be of this kind, say because you can only guarantee that the vaccine will be sterile for a couple of minutes in open air.

2 Method

This deliverable is capable of solving instances of the latter kind (those with only no-wait constraints). Given such a job shop instance, described in an input file, the software computes the optimal schedule and outputs it on the screen. But before we are able to describe how it works, we have to introduce some more notation. Let the term *job* denote here a maximal collection of operations which are connected by no-wait constraints. Here the fixed amount of time can also be negative. In this case the second operation begins before the first one ends. Such a situation occurs for example in liquid processing industries, etc.

Given such a job shop instance we construct a weighted complete directed graph G. Its node set is the set of jobs in the instance plus one dummy node. Its arc weight function is defined as follows. Let (u, v) be an arc of G, and assume

⋆ This work was partially supported by the IST Program of the EU under contract number IST-1999-14186 (ALCOM-FT).

that neither u nor v is the dummy node. We can schedule the job which is associated with u first and draw the corresponding Gantt chart. We then can move v into the chart from the right hand side as far as possible. Then the length of the arc (u, v) is defined to be the additional amount of time needed to execute v after u in this two-job schedule.

If u equals the dummy job, we give the arc (u, v) a length equal to the amount of time needed to process v in an empty environment (=no other jobs given). Note that this is well-defined as the jobs here can only have no-wait constraints. If v equals the dummy node the arc (u, v) gets length zero.

By construction we have the following property

Theorem 1. *The value of an optimal ATSP tour in G is a lower bound on the makespan of the job shop instance by which we constructed G.*

Proof. See [1] □

We solve this ATSP instance by a formulation as an ILP and a branch and cut approach which is due to Dantzig, Fulkerson and Johnson. For details on how the algorithm works you might want to look at the well documented source code.

3 An Example

The example discussed here is not intended to be an interesting one nor a big one. Its purpose is to give an impression of how to call the program and the output the program gives when it is run. You find the listing of the example in Section 5. This example defines a tiny no-wait job shop instance of two jobs. One is having two operations; one on machine one of length five and the other one on the second machine of length 20. The second job has only one operation of length ten on the first machine. Every operation is supposed to start when the whole job starts. Once the program is installed and compiled (see the README file which is delivered with the program how to do this) enter the following command from the directory `[SYMPHONY-HOME]/tsp` to run the program on the example

```
bin.[OS-TYPE]/master_tm_lp_cg_cp -f cfg -F example.jsp,
```

where `cfg` is a configuration file (it is delivered with the application) and `example.jsp` is the file describing the job shop instance. When the program terminates you should see the sequence how to schedule the jobs (in the sense of the definition of the distances of the graph) on the screen.

4 How to Get the Software

Download it at `http://www.cs.uu.nl/people/peterl`. For instructions on how to compile and run this software see the README file that is included in the package.

5 The Example Code

The listing of the example discussed in Section 3.

```
MAX_NUMBER_OF_MACHINES 2
NUMBER_OF_JOBS 2
JOB 1
{
        NUMBER_OF_OPERATIONS    2
        OPERATION
        {
                RELEASE_DATE    0
                DUE_DATE        200
                MACHINE         1
                PROCESSING_TIME 5
                STARTING_TIME   0
        }
        OPERATION
        {
                RELEASE_DATE    0
                DUE_DATE        200
                MACHINE         2
                PROCESSING_TIME 20
                STARTING_TIME   0
        }
}
JOB 2
{
        NUMBER_OF_OPERATIONS    1
        OPERATION
        {
                RELEASE_DATE    0
                DUE_DATE        200
                MACHINE         1
                PROCESSING_TIME 10
                STARTING_TIME   0
        }
}
```

References

1. "The Connection Between the No-Wait Job Shop and the TSP", in preparation, `http://www.cs.uu.nl/people/peterl`.