OR5 Project

Oliver Mulder and Sebastiaan Craens

# Constructive heuristic:

The first heuristic is a constructive heuristic, it is the simplest and it is used to make starting solution for the more complex heuristics to improve upon. There are still important decisions to be made here however since different constructive heuristics have the potential send the later greedy heuristic to different local optimums.

The first constructive heuristic we made which we named constructive heuristic 1 went from order 1 through 30 and placed each order at the machine with the least items on it, with the lowest numbered machine being prioritised as a tie breaking rule, this gave a total deadline penalty cost of 1414. This was much lower than we managed to achieve with any other constructive heuristic, it even outclassed several thousand randomly generated schedules.

## Constructive heuristic 1 pseudo code:

Step 0: initial

Start with a completely empty solution and set index o to 1

Step 1: stopping

If index o is greater than the number of orders, stop

Step 2: step

assign order(o) to the machine with the fewest jobs assigned to it, if there is a tie assign the job to the machine with the fewest jobs and the lowest machine number

Step 3: increment

Set order index o to o + 1

# Complex heuristic:

### First heuristic:

The first greedy heuristic made would check the total deadline penalty cost of every possible combination of swapping two items on the schedule. After checking every possibility it would make the new schedule the schedule with the lowest total deadline penalty cost the. It would then repeat until no swap could produce an improvement at which point it outputs the current best solution as the found local optimum.

This heuristic output a schedule with a total deadline penalty cost of 350 after applying it on constructive heuristic 1 which we were quite happy with though there was still one notable flaw with this heuristic, it could only swap two items between machines or within the same machine, no change could be made to the number of jobs given to a machine. This made this complex heuristic especially ineffective when used on a randomly generated constructive heuristic as there is a possibility of one machine being given very few jobs.

### Improved complex heuristic:

To improve the complex heuristics, additions were made to it. On top of checking every possible swap, the algorithm will now also check every possible combination of swapping full ques between machines as well every combination of moving a single order to every other spot it could be for every individual order. This has caused the improvement heuristic to reach more better local optimums than it did before. Since it is a rather complex heuristic we have done our best to optimise the process as best we can. For one thing, made it such that the best solution for any loop of the algorithms tracks which values actually changed and only recalculates those instead of going over every order again to calculate the total delay penalty cost of that schedule.

# Visualisation:

The way we want to visualize our data is via a figure where the x-axis represents time and the y-axis represents machine number. Every machine should have a bar with colours representing what colour should be on the machine at that time and blank spaces representing cleaning and inactivity. The figure will have a legend as we want to keep the program as generic as possible and with a legend two different colours on the figure can represent paint colours that can’t be told apart as easily.