Heuristic Optimization
Implementation exercise 2

Arnau Dillen - arnau.dillen@vub.be

17 May 2017

1 Introduction

For this implementation exercise we implemented two Stochastic Local Search (SLS) algorithms to solve the permutation flowshop scheduling problem (PFSP) for the weighted completion time (WCT) criterion. The first algorithm we implemented is the Ant Colony Optimization (ACO) algorithm which uses the concept of pheromone trails, based on the behaviour of ants in enatur. The chosen variant of this algorithm is the ACO2 variant from Rajendran & Ziegler (2005). For the second algorithm we chose the relatively simple Iterated Greedy Algorithm (IGA) from Pan & Ruiz (2012). In both cases the original algorithm from the paper uses the flowtime criterion rather than WCT, but this is a trivial modification. There are also differences in termination criterion and used local search techniques for both algorithms. We discuss these differences in detail in the next section. All code for these experiments was written in C++.

2 Algorithms

2.1 Ant Colony Optimization

For the first SLS algorithm we chose to implement the second ACO variant from Rajendran & Ziegler (2005). This type of algorithm is based on the way ants use pheromone trails to signal their peers about a promising path.
The algorithm starts by generating an initial solution and performing a local search on this solution. In this implementation, a solution is constructed with the RZ algorithm from implementation exercise 1. For the local search we perform iterative improvement with a best improvement pivoting rule and insert neighbourhood. We chose this setup because we want to start from a good local optimum before running ACO construction.
The solution obtained from this local search is then used to initialize the pheromone trails we will use in our ACO algorithm. The pheromone trails are represented by a n x n matrix that stores trail intensities for every job and their position in a given solution. We denote the trail intensity for job i at position k as τ_{ik}. Trail intensity will be high for a certain job in a certain position if the job has often been seen in the given position for solutions that have been considered by the algorithm. It represents the desire to place a certain job in a certain position. In our case the trail intensities are initialized to $1/F_{best}$ with F_{best} denoting the WCT of the best solution found so far.
When the pheromone trails have been initialized we can start constructing a new solution with the following procedure.

Sample a uniform random number u in the range $[0, 1]$.

If $u <= 0.4$
then choose the first unscheduled job as present in the best sequence obtained so far;

```
else
    if $u <= 0.8$
        then set $T_{ik} = \sum_{q=1}^{k} \tau_{iq}$;
            among the set of the first five unscheduled jobs, as present in the best sequence obtained so far, choose the job with the maximum value of $T_{ik}$;
        else select job $i$ from the same set of five unscheduled jobs by sampling from the probability distribution $p_{ik} = (T_{ik}/\sum_l T_{lk})$, where job l belongs to the set of the first five unscheduled jobs, as present in the best sequence obtained so far.
        Note that when there are less than five jobs unscheduled, then all such unscheduled jobs are considered.
```

After this solution has been constructed we attempt to improve it with a local search. In our implementation we chose to use iterative improvement with a first improve pivot and using the same neighbourhood as our previous local search. We chose a first improve pivot in this case as we want to quickly reach a local optimum. Otherwise the cost of finding a local optimum would outweigh the cost of building an ACO solution. We are more interested in the exploration provided by ACO than finding a good local optimum with local search.
After this local search we use the achieved solution to update our pheromone trails. The trails are updated with the following rule. If job i is placed in position k in the solution from our local search we update the pheromone trail as such: $\tau_{ik}^{new} = \rho \times \tau_{ik}^{old} + (1/(diff_i \times F_{new}))$, with $diff_i =$ (—position of job i in best sequence so far $-k| + 1)^{1/2}$.
Otherwise $\tau_{ik}^{new} = \rho \times \tau_{ik}^{old}$
The parameter ρ is called the trail persistence and represents the magnitude with which the trail intensity decreases when a job/position combination is not visited anymore.
We repeat the ACO construction procedure and improvement until the termination criterion is reached. The termination criterion for this is reached when a run-time of 500 times that of a Variable Neighborhood Descent run on the same machine, under the same load, is reached. It is also possible to provide a target WCT we want to reach as termination criterion. This is useful for investigating the required run-time for reaching a certain solution quality.

2.2 Iterated Greedy Algorithm

For our second SLS algorithm we decided to implement the iterated greedy algorithm proposed by Pan & Ruiz (2012). This algorithm consists of removing a certain amount of jobs from an initial solution and inserting them again at some position that minimizes the WCT. This algorithm was chosen for its relative simplicity in comparison to the previous ACO algorithm and other algorithms. Our experiments should show us if the increased complexity of ACO results in better performance (for both run-time and solution quality) than the simple IGA.
The initial solution from which we start is constructed in the same way as in our previous ACO algorithm. We first construct a solution with the RZ heuristic and improve it with iterative improvement until a local minimum is reached. As before we use a best first pivot rule and exchange neighborhood for our iterative improvement algorithm.
After generating an initial solution we start a loop that iterates over the current best solution until it either reaches a given run-time or solution quality. This loop consists of two phases called destruction and construction.
In the destruction phase we randomly select a number of jobs from the current solution and remove them. The number of jobs that are removed from a solution in the destruction phase depends on the parameter d that is provided at the start of a run.
The construction phase will reinsert the removed jobs in the partial solution to reconstruct a new complete solution. The position where a job is reinserted depends on a greedy heuristic. A job is inserted at the position where the partial WCT is minimized for the new partial solution.
After construction a new local search is applied in the form of an iterative greedy run with first improve

pivot rule and insert neighborhood. This setup allows us to compare our two implemented algorithms under similar circumstances.
To accept a solution from our local search we use the simulated annealing acceptance criterion. If the computed solution is not better than the currently accepted solution, we might accept it according to the following rule. We start by generating a random number between 0 and 1. We accept a worsening solution if this random number is smaller than $e^{(\sum C_j(\pi) - \sum C_j(\pi'')) / Temperature}$ where $\sum C_j$ denotes the WCT for a solution. The solutions π and π'' represent the current accepted solution and computed solution respectively. Note that the currently accepted solution is not the same as the best solution so far (denoted as π^*). The currently accepted solution can be worse than the best so far because of simulated annealing. Simulated annealing needs a temperature too. This temperature is calculated as follows: Temperature $= \lambda . \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij}}{10mn}$ where λ is a parameter, m and n are the number of machines and jobs respectively, and p_{ij} is the completion time of job j on machine i.

3 Experiments

Regrettably the code for our experiments could not be finished in time. Therefore there are no results available for discussion.

3.1 Average percentage deviation

3.2 Run-time distributions

4 Conclusion

References

Pan, Q.-K., & Ruiz, R. (2012). Local search methods for the flowshop scheduling problem with flowtime minimization. *European Journal of Operational Research*, *222*(1), 31–43.

Rajendran, C., & Ziegler, H. (2005). Two ant-colony algorithms for minimizing total flowtime in permutation flowshops. *Computers & Industrial Engineering*, *48*(4), 789–797.