Large scale optimization (TMA521) Project 2

Machining planning problem

Gustav Hjelmare
880715-2452
hjelmare@student.chalmers.se

Albin Karlsson
920210-0138
albinka@student.chalmers.se

Abstract

In this project we have used Dantzig-Wolfe decomposition and column generation method to solve a job scheduling problem. The program that was written worked well for problems of smaller sizes (15 jobs) since the column generation succeeds in creating useful schedules. In larger cases (30 jobs), the program fails to generate useful schedules, so that only the initial heuristically created schedule is feasible. Thus, the algorithm as implemented here does not outperform the heuristic at this problem size.

March 10, 2015

1 Introduction

The aircraft industry is highly complex, and this complexity extends down to the scheduling of machining operations performed on aircraft parts. Here, we consider one such scheduling problem.

Individual parts are to be processed in various ways, and some of this processing happens in a so called multitask (MT) cell. This MT cell consists of several machines, each of which can carry out some specific operations. When a part arrives at the MT cell, it is first mounted in a fixture by one machine then sent to another machine for some machining operation. After the main machining operation, there may be a need for manual and/or automatic deburring, and finally demounting of the part from the fixture, before it can leave the MT cell and continue on its way through the factory. It is the scheduling of which part should be in which stage of this process at a certain time that constitutes the full problem under consideration.

In practice, it turns out that the mounting/demounting and manual/automatic deburring machines in the MT cell are not very heavily used, but the five main machining machines constitute a bottle neck in terms of throughput. Thus, it is sensible to consider the scheduling of machining operations separately. After finding a good solution to that problem, one can then adjust that solution slightly in order to accommodate mount/demount and manual/automatic deburring operations. At the time of writing, we have not yet had the time to consider the adjustments needed, so the remainder of this report will deal only with the scheduling of machining operations, not with pre- and post-processing schedules.

The scheduling of machining operation (hereafter referred to as the machining problem) consists of finding a schedule that finishes as many jobs as possible as early as possible, while respecting that some of the machines have limitations in terms of what they can do (e.g. they may only work with certain materials, thus excluding some parts from being processed by that particular machine).

2 Computational method

In this report we will not define the original full problem (see task paper if you would like more information about that), we will only present the machining problem that was separated from a larger job-planning problem. A mathematical formulation of the machining problem is stated below:

$$\text{minimize} \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{K}}\sum_{u\in\mathcal{T}} \left(A_j(u+p_j^{pm}) + B_j \cdot max\{u+p_j^{pm}-d_j],0\}\right) x_{jku} \quad (1)$$

$$\sum_{k\in\mathcal{K}}\sum_{u\in\mathcal{T}} x_{jku} = 1, \qquad \text{j} \in \mathcal{J}$$

$$\sum_{u\in\mathcal{T}} x_{jku} \le \lambda_{jk}, \qquad \text{j} \in \mathcal{J},\ k\in\mathcal{K} (2)$$

subject to $$\sum_{j\in\mathcal{J}}\sum_{v=max\{u-p_j+1,0\}}^{u} x_{jkv} \le 1, \qquad \text{k} \in \mathcal{K},\ u\in\mathcal{T} (3)$$

$$\text{x}_{jku} = 0, \qquad \text{j} \in \mathcal{J},\ k\in\mathcal{K},\ u = 0{,}1{,}...{,}(a_k-1) (4)$$

$$\text{x}_{jku} \in \{0{,}1\} \qquad \text{j} \in \mathcal{J},\ k\in\mathcal{K},\ u\in\mathcal{T} (5)$$

In this description, j is a job number, k denotes a machine, and u is a timestep number, with $\mathcal{J}$, $\mathcal{K}$ and $\mathcal{T}$ being the corresponding sets. The x_{jku} is 1 if job j is started on machine k at timestep u and 0 otherwise, and thus constitutes the schedule that we seek to optimize. In order to do this, the machining problem is deconstructed into two parts. The first part uses some set of existing schedules, and finds a good combination of them. The second part uses the dual variables from the first part, and generates new schedules, which can then be fed back to the first part for further iterations.

2.1 Restricted master problem

The restricted master problem can be seen below.

minimize $$\sum_{k\in\mathcal{K}}\sum_{\ell\in\mathcal{L}_k}\left(\sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{T}}\Big(A_j(u+p_j^{pm})+B_j\cdot max\{u+p_j^{pm}-d_j],0\}\Big)x_{jku}^{\ell}\right)\tau_k^{\ell} \tag{6}$$

subject to
$$\sum_{k\in\mathcal{K}}\sum_{l\in\mathcal{L}_k}\Big(\sum_{u\in\mathcal{T}} x_{jku}^{l}\Big)\tau_k^{l} = 1, \qquad \text{j} \in \mathcal{J}(7)$$

$$\sum_{\ell\in\mathcal{L}_k}\tau_k^{\ell} = 1 \qquad \text{k} \in \mathcal{K}(8)$$

$$\tau_k^{\ell} \ge 0, \qquad \ell\in\mathcal{L}_k,\ k\in\mathcal{K}(9)$$

$$\tau_k^{\ell} \in \{0{,}1\}, \qquad \ell\in\mathcal{L}_k,\ k\in\mathcal{K}(10)$$

The restricted master problem is used in ampl to both gain the dual variables (π_j and γ_k) and the parameter τ that tells us what schedules that should be used.

2.2 Column generation sub problem

The column generation problem for each machine (k) can be seen below.

$$\text{minimize} \sum_{j\in\mathcal{J}}\sum_{u\in\mathcal{T}} \left(A_j(u+p_j^{pm}) + B_j \cdot max\{u+p_j^{pm}-d_j],0\} - \pi_j\right) x_{jku} - \gamma_k \tag{11}$$

subject to

$$\sum_{u\in\mathcal{T}} x_{jku} \leq \lambda_{jk}, \qquad j \in \mathcal{J} \tag{12}$$

$$\sum_{j\in\mathcal{J}} \sum_{v=max\{u-p_j+1,0\}}^{u} x_{jkv} \leq 1, \qquad u \in \mathcal{T} \tag{13}$$

$$\text{x}_{jku} = 0, \qquad j \in \mathcal{J},\ u = 0,1,...,(a_k-1) \tag{14}$$

$$\text{x}_{jku} \in \{0,1\} \qquad j \in \mathcal{J},\ u \in \mathcal{T} \tag{15}$$

This problem description is used to gain the new columns, i.e. the solution x^{ℓ}_{jku}.

2.3 Heuristic

The column generation subproblem 11 depends on the dual variables of the restricted master problem 6, and the restricted master problem can only be solved by using some set of columns to combine. Thus, in order to start iterating, one has to have some initial set of columns. We create this initial set by a simple heuristic, in which we look through the jobs in order, and simply schedule them at the first available time slot on the first machine that can perform the job in question.

2.4 Exit criteria

When performing a column generation method one adds columns to the master problem as long as the solution to the master problem gets better given more columns. The reduced cost c_r is the parameter that tells us if we would benefit from considering more columns. Since we want to minimize the master problem we optimally want to stop the iterations as soon as the reduced cost is non-negative. This is exactly what is done in the problem of 15 jobs that was solved using our program. Note that although *optimally* we would stop iterating when the reduced cost goes non-negative, this might take too much time and one should consider adding an exit criteria based on

the distance between upper and lower bounds.
The reduced cost is computed as in equation 2.4 and the upper and lower bound is shown in equation 2.4 and 2.4.

$$c_r^{(k)} = \min_{\pi,\gamma} \sum_{j\in\mathcal{J}} \sum_{u\in\mathcal{T}} \left(A_j(u+p_j^{pm}) + B_j \cdot max\{u+p_j^{pm} - d_j], 0\} - \pi_j\right) x_{jku} - \gamma_k \tag{16}$$

$$UBD = \min_{x\in X} \sum_{k\in\mathcal{K}} \sum_{\ell\in\mathcal{L}_k} \left(\sum_{j\in\mathcal{J}} \sum_{u\in\mathcal{T}} \left(A_j(u+p_j^{pm})+B_j \cdot max\{u+p_j^{pm}-d_j], 0\}\right) x_{jku}^{\ell} \right) \tau_k^{\ell} \tag{17}$$

$$\text{LBD} = \text{UBD} + \min_k c_r^{(k)} \tag{18}$$

3 Result

When solving the problem defined by `2010-11-17_15j_MTC6.dat`, consisting of 15 jobs, the relaxed restricted master problem is solved to optimality after 25 iterations, and the columns generated form a better schedule than the inital heuristic had provided. The result during the run of the program can be seen in figure 1, 2 and 3, note that the reduced cost goes to zero as well as that the upper and lower bound converges to the same value.

Doing the same for 30 or more jobs (`2010-08-16_30j.dat` and `2010-08-16_40j.dat`) proved to not work so well. In the end, when solving the master problem *with* the binary constraint on τ, the final result is the same schedule that the initial heuristic produced even though it is clear that it is far from the optimal solution. The result from running 30 jobs in the program can be seen in figure 4, 5 and 6.

4 Discussion

What should be reflected over is why the program performs well for 15 jobs while performing so badly for 30 jobs. We believe that this is because with just 15 jobs solution from the relaxed problem happens to give τ values of 0 and ones with out it being under the binary constraints. In the end of the

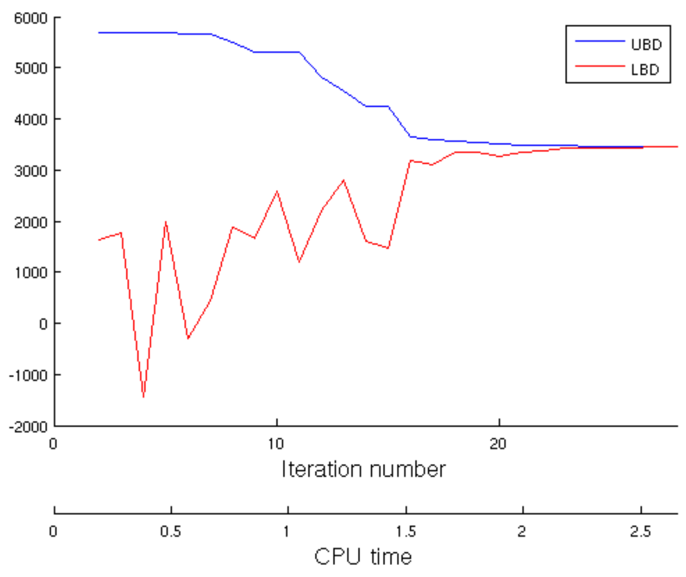

Figure 1: The upper and lower bounds through out the iterations of our program when optimizing the machining problem with 15 jobs. Note that they converge to the same value in the last iterations.

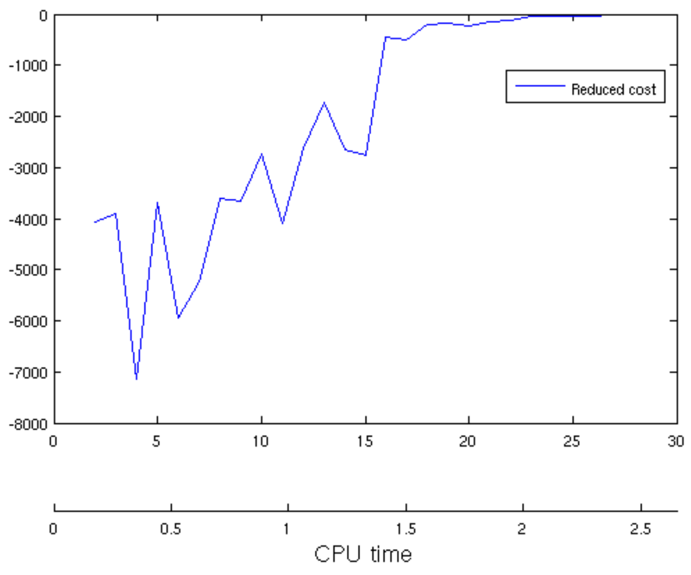

Figure 2: The reduced cost through out the iterations of our program when optimizing for 15 jobs.

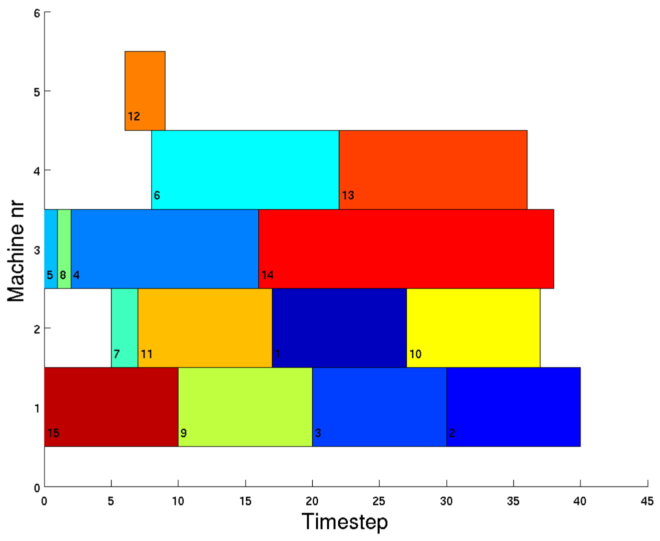

Figure 3: The resulting schedule of the 15 job problem after running the program. The initial unused time on machines 2, 4, and 5 is due to the machines not being available for scheduling at that time.

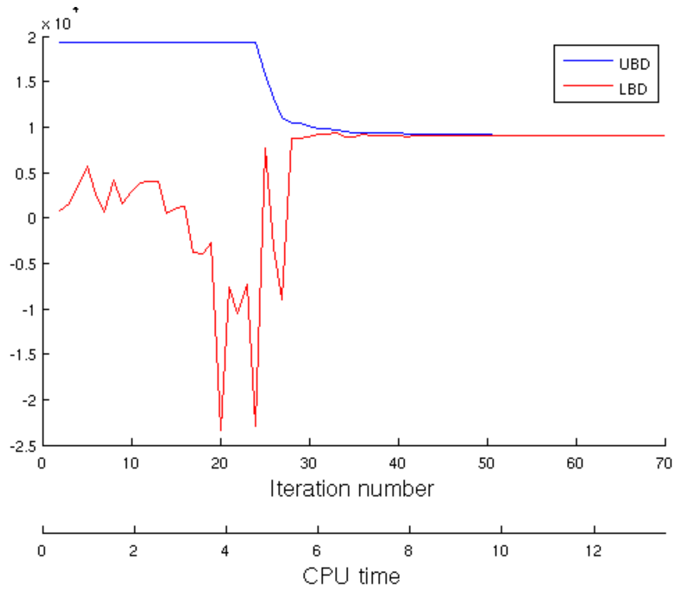

Figure 4: The upper and lower bounds through out the iterations of our program while optimizing for 30 jobs. Note that they converge to the same value in the last iterations.

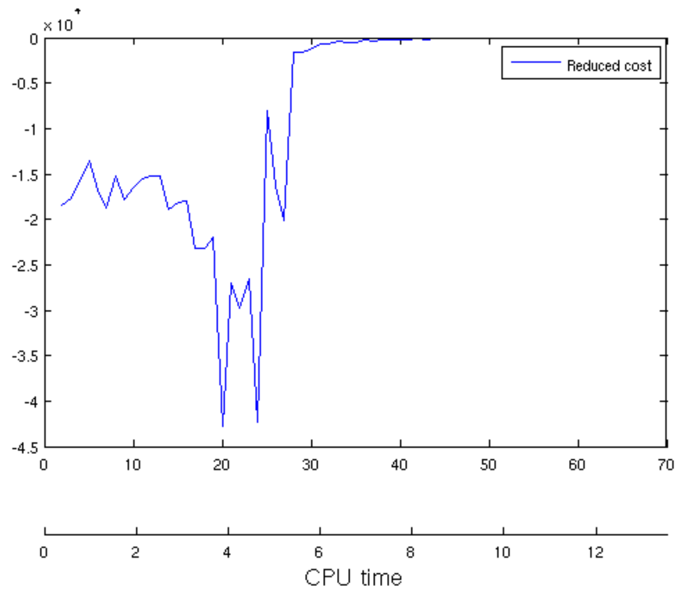

Figure 5: The reduced cost through out the iterations of our program when optimizing for 30 jobs.

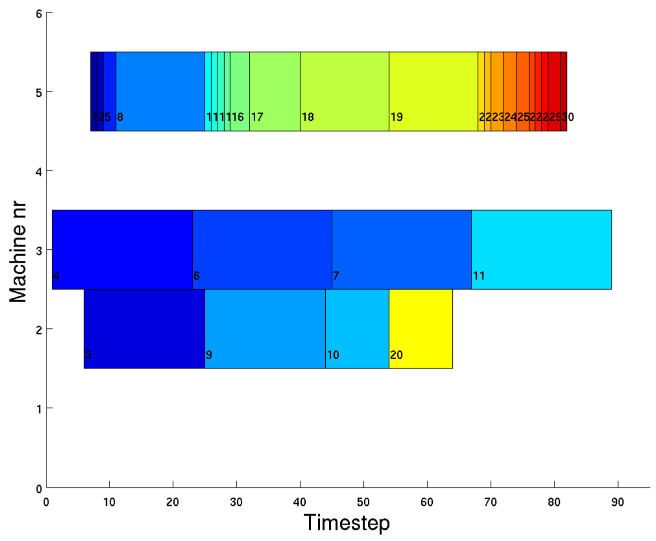

Figure 6: The resulting schedule of the 30 job problem after running the program. The schedule is obviously not the optimum one since there are two machines that are not being used at all. This is consistent with the way our initial heuristic would schedule jobs.

program when one wants to make sure that τ is binary and solves the master problem with this additional constraint it is to no surprise that this can be done without any complications. On the other hand when working with a higher number of jobs the program has been unable to generate columns such that τ can satisfy the binary constrant, resulting in mixtures of schedules. In this case, when we want τ to abide by the binary constraint it so happens that it can not find any feasible set of schedules that was constructed throughout the program, except the feasible set that was constructed by the heuristic. We believe that the program should be able to be improved by also including branch and bound methods.