A lower bound and a tabu search algorithm for scheduling batches in a proportional job shop *

Márton Drótos † Gábor Erdős ‡ Tamás Kis §

1 Introduction

We consider a complex batch scheduling problem in a job shop, where jobs belong to job families, there are sequence and machine dependent setup times between jobs of different families, the processing times are determined by the job sizes and machine speeds, there are routing and machine alternatives and the machines have non-availability periods. The objective is to minimize the total tardiness of the job families, where the tardiness of a job family is determined by the maximum tardiness of the jobs in the family. This is a complex problem motivated by an industrial application. Nevertheless, we are able to compute lower bounds on large scale instances, and some ideas of our tabu search algorithm may be transfered to other batch scheduling problems. For convenience we measure the time in minutes and days.

The *input data* of our scheduling problem can be summarized as follows: there is a set of *job families* $\{F_1, \ldots, F_n\}$, where each F_i consists of a finite number of jobs. All jobs in the same family F_i have a common *release date* r_i, *due date* d_i and *weight* w_i. There is a *time horizon*, 30 days, say, and each r_i falls on the beginning of some day, while each d_i is the end of some day within the horizon. Moreover, each job $j \in F_i$ has a size q_j. There is a sequence of *main production steps* and each job family requires a subsequence of this. Each F_i has a few *routing alternatives* $R_i^1, \ldots, R_i^{a_i}$, where each R_i^ℓ is a sequence of stages, each *stage* being a subsequence of steps. The stages of each routing alternative must be disjoint and their union must give the set of steps required by F_i. With each stage s, there is associated a set of machines $\mathcal{M}_i^s$. The *processing time* of job $j \in F_i$ on some machine $M_k \in \mathcal{M}_i^s$ is defined as $p_{j,k} = q_j/v_i^k$, where v_i^k is the *speed* of M_k (items/unit time) when processing any job from F_i. Each machine M_k has a *calendar* specifying those time periods when the machine is available for processing. There are *sequence dependent setup times* between the jobs of different families scheduled on the same machine. Let $u_k(j_1, j_2)$ denote the setup time between the jobs j_1, j_2 on M_k, it is 0 if jobs j_1 and j_2 belong to the same family. It is assumed that the setup times satisfy the triangle inequality, that is, $u_k(j_1, j_2) + u_k(j_2, j_3) \geq u_k(j_1, j_3)$ for any jobs j_1, j_2 and j_3.

A *feasible solution* consists of a selection of routing alternatives $R_i^{\rho(i)}$, *starting times* $S_{j,s} \geq r_i$ ($j \in F_i$, $s \in R_i^{\rho(i)}$), and *machines* $\mu(j, s) \in \mathcal{M}_i^s$, $i = 1, \ldots, n$. Notice that the same stage of distinct jobs of the same family may be assigned to different machines. If a machine becomes non-available during the processing of a job, then the processing of the job is stopped and resumed immediately once the machine becomes available again. However, once the processing of a job has started on a machine, it must be finished on the same machine, and it cannot be preempted for processing another job. Before processing

*The support of NKFP Grant No. 2/010/2004 and the János Bolyai Research Grant No. BO/00380/05 is gratefully acknowledged.

†marton.drotos@sztaki.hu. Computer and Automation Institute, Kende utca 13-17, 1111 Budapest, Hungary, and Budapest University of Technology and Economics, Magyar tudósok körútja 2/d, 1117 Budapest, Hungary.

‡gabor.erdos@sztaki.hu. Computer and Automation Institute, Kende utca 13-17, 1111 Budapest, Hungary.

§tamas.kis@sztaki.hu. Computer and Automation Institute, Kende utca 13-17, 1111 Budapest, Hungary.

the first job of a family on a machine, a setup must be performed. In addition, all jobs of the same family that are assigned to the same machine must be performed consecutively. The objective is to minimize the function $\sum_i w_i \lceil C_{F_i} - d_i \rceil$, where C_{F_i} is the maximum completion time of the jobs in family F_i and $\lceil \cdot \rceil$ rounds up to days.

2 Solution approach

Our solution approach consists of two main phases. In the first phase a routing alternative is selected for each job family and an initial (job, stage) to machine assignment as well as an initial ordering of jobs is determined. In the second phase we apply a tabu search algorithm for improving the initial schedule.

2.1 Selection of routing alternatives and lower bound computation

We select a routing alternative simultaneously for all the job families. To this end, we allow preemption and neglect setup times and exploit that the processing times of the jobs depend on the job sizes and machine speeds. We also exploit that all jobs of the same family go through the same stages. In the relaxed problem, we take into account the machine availability periods and job family due dates. Like in the original problem, the objective is to minimize the total weighted tardiness of job families, where tardiness is measured in days. We provide a time-indexed MIP formulation and solve it by a standard solver. The advantage of this approach is that once the MIP solver finds a feasible, but not necessarily optimal solution, we can stop at any time, we always have a lower bound on the optimal objective function value.

The solution of the MIP problem provides not only a selection of routing alternatives, but also a fractional assignment of job families to machines. From this we create an initial schedule by assigning full jobs to machines and by inserting the batches (formed by all jobs of the same family assigned to the same machine) into an initially empty schedule in a greedy manner.

2.2 Tabu search

To improve the initial schedule, we have used a tabu search algorithm. The *neighborhood* of a schedule is defined by feasible reinsertions of critical batches, where a batch of jobs is *critical* if moving it to the left would enable jobs that complete after their due dates to finish earlier. Note that since the number of batches is much smaller than the number of jobs, this way we have reduced the size of the search space considerably. To define various neighborhoods of a schedule, we define a critical forest of batches, which is any spanning forest of the subgraph spanned by the critical batches. In neighborhood N_1, one critical tree of the critical forest is chosen at random and all batches of this particular tree are tested for reinsertion on all possible machines and the best neighbor is chosen. Neighborhood N_2 is similar to N_1, but if the objective function value improved in the previous iteration, then it evaluates the trees of the critical forest one by one until an improvement is found, otherwise only one randomly chosen tree is evaluated. N_3 is similar to N_2, but if the previous iteration improved the objective function value, then the evaluation starts with the tree of the same root from which a batch was chosen for reinsertion in the previous iteration (if such a tree exists). To guide the search we rank the solutions with respect to two objectives: total tardiness of job families as primary objective and total job tardiness as secondary objective. The secondary objective helps to find out how to improve on the primary objective. On top of that, if there is no improvement after some iterations, then we split some of the critical batches (provided it improves the objective function) and the tabu search continues with the modified set of batches. *Splitting a batch* means that some jobs of the batch are moved to a different machine.

We tested our algorithm on data from a real-world industrial application. Results show that batch splitting considerably improves solution quality. Moreover, there are not big differences between the neighborhoods.