Scheduling multipurpose batch process industries with no-wait restrictions
ALCOM-Deliverable D9, WP 3

Han Hoogeveen
Institute of Information and Computing Sciences
Utrecht University
P.O. Box 80089, 3508 TB Utrecht, The Netherlands
Email: slam@cs.uu.nl

June 1, 2001

Abstract

Scheduling problems in multipurpose batch industries are very hard to solve because of the job shop like processing structure in combination with the rigid technical constraints, such as no-wait restrictions. In this note, we present an outline of the algorithmic approach that we intend to follow to tackle these problems.

1 Introduction

In this report we describe our algorithmic approach to solving a number of novel scheduling problems that occur in the pharmaceutical industry. The pharmaceutical industry is an example of a multipurpose batch process industry. This is a special type of process industry in which products may follow different routings through the plant. Generally a variety of different products is produced on a set of multipurpose machines. Often products may be produced on one of several similar or identical machines. Multipurpose batch process industries are preferred for low volume products, such as specialty chemicals and pharmaceuticals, because of their inherent product flexibility. This copes with the current trend of increasing product variety and decreasing demand volumes for individual products.

Scheduling problems in multipurpose batch process industries are very hard to solve because of the job shop like processing structure in combination with rigid technical constraints, such as no-wait restrictions. There are three essential aspects of the scheduling problem. First, different operations belonging to the same job may need to be carried out without any waiting time allowed, because the intermediate product is unstable. Second, operations for the same job may have to be performed at the same time at a number of machines. This means that operations of a job may need to be overlapping in time. Third, several parallel machines may be available to perform a specific operation. An additional complication is that once-in-a-while a chemical reaction fails, which implies that this step has to be repeated, and hence the schedule has to be adjusted.

2 The scheduling model

The basic model for production scheduling problems is the job shop scheduling problem. The job shop problem is defined as follows. There are n jobs $1, 2, \ldots, n$ and m machines $M_1, M_2, \ldots, M_m$; each machine is continuously available from time zero onwards and can process no more than one job at a time. Each job j consists of n_j operations $O_{1j}, O_{2j}, \ldots, O_{n_j,j}$, where operation O_{ij} has to be processed on a given machine (not necessarily M_i) for p_{ij} time units. Operation O_{ij} can only start if the preceding operation $O_{i-1,j}$ in job j has been completed. The problem is to determine a starting time (or equivalently, an execution interval) for each operation such that the above constraints are met. The objective is to minimize the time at which the last job is completed.

This basic problem can be 'solved' quite efficiently through local search methods, like simulated annealing and tabu search (Vaessens et al., 1996). Unfortunately, the characteristics of our problem make the basic job shop problem insufficient, since it does not account for parallel machines, no-wait restrictions, and overlapping operations. Hence, we add these characteristics to our job-shop scheduling problem. The restriction of identical, parallel machines can be dealt with by requiring that at each time at most a given number of operations make use of a machine of a certain type; the remaining constraints can be modelled by a straightforward relation on the starting times. The objective of minimizing the makespan is okay, since it is common practice to split up the work in batches that have to be dealt with in the upcoming four weeks.

In Raaymakers and Hoogeveen (2000), we have applied the traditional local search approach on this extended job shop problem. Although the obtained results were consistently better than the results obtained in practice through some simple heuristics, we are not fully satisfied. In comparison to the basic job shop problem, our local search method does not work very well, as a small change in the current schedule cannot be implemented efficiently, because of the no-wait constraints. Hence, a part of the schedule has to be computed from scratch again when studying a neighbor of the current solution, which implies that the 'small' change in the local search method has become a 'big' change. Therefore, we want to apply another method, called *constraint satisfaction.*

3 Constraint satisfaction

A recent development in scheduling theory is to use constraint satisfaction techniques in job shop scheduling. The idea behind applying constraint satisfaction is to specify for the starting times of each operation a *domain* of possible values such that no operation starts before time zero and the last operation is completed at a given upper bound. The domains are then tightened by applying logical combinations of the constraints, where the consequences of a successfull domain reduction are used to update the other domains through *constraint propagation.* We refer to Dorndorf, Pesch, and Phan Huy (2000) for an overview of the application of constraint satisfaction techniques in job shop scheduling.

Especially in our case, with the no-wait relations between the operations, constraint satisfaction seems a suitable approach, since the propogation step can be expected to be powerful. The approach is as follows. First, we tighten the domains as much as possible by applying fast *consistency rules.* A consistency rule checks, given that some operation j is assigned to some starting time S from the domain of operation j, whether it is still possible to find a feasible solution; if this is not the case, then this S can be removed from the domain of operation j. Devising strong consistency rules with a low polynomial running time is a hot topic; no research has been reported so far on the special case with no-wait restrictions. After this phase, we may expect that the operations can be divided in *clusters* that consist of a number of jobs, where a cluster is defined such that the starting time of one operation of a job in the cluster specifies the starting times of the other operations in the cluster to a large extent; this implies that the cluster as a whole can be moved in time, but any separate operation in it cannot. Given these clusters, we can then specify a number of possible outline schemes, which embed the relative position of the clusters in a schedule; in fact, we are partitioning the resulting solution space in a number of subsets, which are described by the underlying cluster-structure. We then want to design a local search algorithm that can quickly implement small changes in the current solution, where the current structure is kept. One of the fundamental questions in this respect is whether we will be able to search the whole solution space of a subset.

A further advantage of constraint satisfaction is that the results of the domain reduction step can still be used to some extent when a chemical reaction fails and the corresponding operation has to be repeated. The bottom line is that, given failure probabilities, we can strive towards finding a schedule that is as insensitive as possible to disturbances.

When we have derived a working prototype, we expect to test it on real-life instances that we obtain from the pharmaceutical company Diosynth.

Acknowledgements

This research was partially supported by EC Contract IST-1999-14186 (Project ALCOM-FT).

References

[1] U. Dorndorf, E. Pesch, T. Phan Huy (2000). Constraint propogation techniques for the disjunctive scheduling problem. *Artificial Intelligence 122,* 189-240.

[2] W.H.M. Raaymakers, J.A. Hoogeveen (2000). Scheduling multipurpose batch process industries with no-wait restrictions by simulated annealing. *European Journal of Operational Research 126*, 131-151.

[3] R.J.M. Vaessens, E.H.L. Aarts, J.K. Lenstra (1996). Job shop scheduling by local search, *INFORMS Journal on Computing 8,* 302-317.