Towards an industrial manufacturing scheduling problem and test bed

W. Nuijten, T. Bousonville, F. Focacci, D. Godard, and C. Le Pape

ILOG S.A., Gentilly, France.
e-mail: nuijten,tbousonville,focacci,dgodard,clepape@ilog.fr
url: http://www.ilog.com

Keywords. manufacturing, test bed, industrial scheduling, setup times, setup costs, earliness, tardiness, calendars, breaks, productivity profiles.

1 Introduction

In this paper a manufacturing scheduling problem is introduced inspired by the industrial manufacturing scheduling problems the authors encountered over the years. In addition to this a set of instances of the problem is presented, which are in turn inspired by industrial instances the authors came across. This set of instances is a first step towards creating MaScLib (MaSc standing for **Ma**nufacturing **Sc**heduling), a test bed for manufacturing scheduling. The goal of this test bed is to facilitate manufacturing scheduling research by providing an industrial basis to test scheduling algorithms and through that increase overall interest in manufacturing scheduling research.

Scheduling the manufacturing of different types of products using a shared set of resources is a difficult task as both finding a *feasible* schedule as finding a *good* schedule, i.e., a schedule of satisfactory quality, can be hard. A feasible schedule for instance must satisfy a wide variety of constraints like temporal constraints, resource assignment constraints, capacity constraints, setup constraints, etc. Finding a good schedule involves responding to a number of conflicting optimization criteria like fulfilling all or at least the most important production orders, delivering the products on time, i.e., not too late to avoid customer dissatisfaction nor too early to minimize storage cost, manufacturing the products using the most cost effective resources, avoiding long and costly machine setups, etc.

In Section 2 of this paper a manufacturing scheduling problem is defined that captures a fair amount of the abovementioned properties. Section 3 describes the current state of MaScLib, after which Section 4 presents some conclusions and future directions.

2 The Problem

This section informally describes the main concepts of the manufacturing scheduling problem we propose to study. Given are a set of *activities* and *resources*. An activity represents the production of a given quantity of a given (intermediate) product. Activities can be linked by *temporal constraints* linking the start time or end time of two activities and possibly specifying a minimal or maximal delay between these times. One or several *due date constraints* can be attached to each activity. A due date constraint specifies either a date at which the activity shall start or a date at

which the activity shall be finished. Four parameters are associated with each due date constraint: the *earliness fixed cost efc*, the *earliness weight ew*, the *tardiness fixed cost tfc*, and the *tardiness weight tw*. Now, if an activity is early by x units of time, the total cost incurred will be $efc + ew * x$ and if the activity is late by y units of time, the total cost incurred will be $tfc + tw * y$. When several due date constraints are associated with the same activity, the maximal value of the corresponding earliness or tardiness costs is incurred.

To each activity a *setup type* and a set of *modes* are associated, where each mode specifies a *primary resource* able to perform the activity, the *capacity* (i.e., the number of units) required from this resource, a *calendar* under which the activity is executed, a set of *secondary resources* and required capacities for each of these resources, the *mode cost* of the mode, and minimal and maximal start times, end times, and processing times for the mode. In a solution each activity will be assigned a mode, but one can decide that an activity will be *unperformed*, meaning that the activity will not require capacity of the primary and secondary resources in the mode. Note that the activity then will obey the calendar of the mode, obey eventual temporal constraints, etc. For each mode an activity has a positive *non-performance cost* that is incurred if the activity is unperformed and assigned to that mode. It can also be declared that an activity has to be performed. One can have a *performance compatibility constraint* between two activities expressing that either they are both performed or they are both unperformed.

A *calendar* defines the execution conditions of a mode and consists of a list of breaks and a *productivity profile*. An activity in a given mode can be interrupted by breaks smaller than the maximal break duration mBD of the mode. An activity is thus *breakable* when $mBD > 0$ and *not breakable* when $mBD = 0$. The productivity profile defines the efficiency of the activity execution. If an activity is scheduled in a time interval with productivity $p\%$, $p\%$ of a processing time unit is executed per time unit. The processing time of an activity is therefore equal to the integral, from start to end, of the productivity.

Each *resource* has a given *capacity profile* expressing that at any point in time, the sum of the capacities required by the activities performed by the resource cannot exceed the resource capacity. A *time origin* is specified for each resource. No activity can be executed on the resource before the time origin.

Setup times and *setup costs* can be defined on resources of capacity 1, also called *machines*. These setup times and costs are defined based on setup types. So if the setup time from type τ_1 to τ_2 is t, it means that between the end of each activity of setup type τ_1 and the start of a consecutively scheduled activity of setup type τ_2 at least a time t must elapse. In fact one can express a more complicated case where the setup time is governed by a calendar in a similar way as for activities. In turn, a setup cost of c from type τ_1 to τ_2 means that a cost c is incurred for each pair of consecutively scheduled activities of type τ_1 followed by type τ_2. An *initial setup type* and a *final setup type* can also be given. The initial setup type specifies under which setup type the resource operated just before its time origin. Note that between the time origin of a resource and the start of any activity, the setup time from an eventual initial setup type to the activity is required. The final setup type specifies under which setup type the resource must be restored at the end of the schedule.

The problem is now to for each activity choose a mode, decide whether or not to perform it, and to schedule it in time such that all constraints are satisfied and the optimization criterion is minimized. The optimization criterion is a linear combination of the *total mode cost* of the chosen modes of all activities, the *total non-performance cost* of all unperformed activities, the *maximal* and *total earliness* and *tardiness cost* of all activities, and the *total setup cost* being the sum over all resources of the initial setup cost, the intermediate setup costs between any pair of consecutive activities, and the final setup cost on the resource.

3 The MaScLib

A test bed containing only instances having the full complexity of the problem presented in Section 2 would obviously be impractical. We thus decided to define different problem class "levels" that gradually get more complex. The first levels do include the features that are the most important from an industrial point of view. Less important features are introduced only at higher levels. Like this we aim to make it relatively easy to do research on problem classes of the first level, while preserving the property that the research is industrially relevant.

For each problem class we will provide a set of instances. An instance is identified by a triple PSI, where P represents a problem class, S represents a typical problem size, and I represents an instance number. Each instance can be used with different combinations of optimization criteria. We consider four combinations to be particularly interesting. The first is to optimize the sum of the mode, setup, and tardiness costs, where all activities have to be performed. The three others are obtained by adding the earliness costs, the non-performance costs, and both.

Problem class level 1 concentrates on two features which are the most frequently found in industry, *i.e.*, temporal constraints and calendars. All problem classes in this level have no setups, at most one due date per activity with $efc = tfc = 0$, a unique mode per activity, no secondary resources, and no capacity profiles. The simplest problem class, called "No-Calendar One-Step" (NCOS) has no calendars, no temporal constraints, and one machine. The "No-Calendar General-Shop" (NCGS) problem class generalizes this by allowing several machines and an arbitrary set of temporal constraints with both minimal and maximal delays. The "BReak One-Step" (BROS) and the "Break-and-Productivity One-Step" (BPOS) problem classes generalize the NCOS class in another direction by including breaks and productivity profiles. In Nuijten et. al (2003) a definition of the different problem classes is given using (an extended form of) the well-known $\alpha|\beta|\gamma$ notation.

In level 2, setups are introduced. Three problem classes are currently defined: STC-NCOS, STC-NCGS, and STC-BPOS, adding "Setup Times and Costs" to NCOS, NCGS, and BPOS. In level 3, "Multiple Modes" are introduced leading to the three problem classes MM-STC-NCOS, MM-STC-NCGS, and MM-STC-BPOS.

Table 1 provides the typical problem sizes (in number of activities), the maximal CPU time (in seconds) we recommend for each on a 1GHz computer, and the number of instances available for each problem class P and problem size S. Instances with about 10 activities ($S = 0$) are mainly there for debugging purposes. Most instances with 50 activities or less ($S = 1$ or $S = 2$) are subproblems of industrial problems or instances exhibiting special characteristics. Instances with 75 activities or more are mostly derived from industrial problems. For certain instances we transformed an

S	0	1	2	3	4	5	6	7	8	9	Total
Typical size	10	25	50	75	100	250	500	1000	2500	$\geq$ 10000	
Recommended CPU time	60	150	300	450	600	1200	1800	2400	3600	7200	
NCOS	10	10		4	2	2	2				30
NCGS		4	2	2	4	10		2	2		26
BROS	4	2			2						8
BPOS	2										2
STC-NCOS	2	2		4	2	2	2				14
STC-NCGS			2	6		4					12
STC-BPOS		2			2						4
MM-STC-NCOS		4	2								6
MM-STC-NCGS				2	2						4
MM-STC-BPOS			2								2
Total	18	24	8	18	14	18	4	2	2		108

Table 1. Currently available instances in MaScLib

industrial instance by removing side constraints and rescaling times and costs, this while preserving the combinatorial essence of the instance. A few instances are randomly generated instances provided by the scientific community. The current total represents 108 instances times the four optimization criteria combinations, hence 432 individual problems. We aim to quickly increase the number of available instances. The instances can be obtained at `http://www2.ilog.com/masclib/`. They have already been provided to several people in the scientific community. Danna (2003) has used the NCGS instances with the four optimization criteria combinations to test an extended disjunctive MIP model. Savourey et. al (2003) plan to use the NCOS instances to test a tabu search method to minimize tardiness.

4 Conclusion

We defined a manufacturing scheduling problem inspired by practice and presented the first version of MaScLib, a set of manufacturing scheduling instances aimed to become an industrial test bed for scheduling algorithms.

In the future we aim to extend MaScLib with instances that have *mode compatibility constraints* to express things like process routes, *setup activities* to express the need for resources when doing a setup, *reservoirs*, i.e., resources for which capacity can be produced and from which capacity can be consumed, *batch resources*, i.e., resources on which activities cannot execute in parallel unless they have the same setup type and start and end at the same time, etc. Corresponding problem classes will be introduced. For all new and existing problem classes we will concentrate on obtaining larger instances than currently present in MaScLib.

References

DANNA, E. (2003): Improving and extending the disjunctive MIP model for the earliness-tardiness job-shop scheduling problem. *Submitted to PMS'2004.*

NUIJTEN, W., BOUSONVILLE, T., FOCACCI, F., GODARD, D., and LE PAPE, C. (2003): MaScLib: Problem description and test bed design. `http://www2.ilog.com/masclib/`.

SAVOUREY, D., JOUGLET, A., CARLIER, J., and BAPTISTE, P. (2003): A tabu search method to minimize total weighted tardiness on one machine with release dates. *Submitted to PMS'2004.*