Integrated maintenance scheduling for semiconductor manufacturing

Andrew Davenport
davenport@us.ibm.com
Department of Business Analytics and Mathematical Science,
IBM T. J. Watson Research Center,
P.O. Box 218, Yorktown Heights, NY, 10598, USA

Abstract

We present a maintenance scheduling problem arising from semi-conductor manufacturing which is characterized by low resource contention and multiple complex objectives and preferences. Since semi-conductor manufacturing involves a very high degree of uncertainty at the level of detailed operations, such as machine availability, yield, and processing times, generating a maintenance schedule which takes into account production operations is a challenging problem. We have developed an integrated approach, with respect to production operations, which uses simulation to estimate expected levels of work in process in the fab. We present details of our solution approach which is based on goal programming, constraint programming and mixed-integer programming.

Introduction

Machines in a manufacturing plant require regular maintenance over their lifespan (e.g. cleaning, calibration, safety checks) to keep them running smoothly. In capital intensive industries, such as semi-conductor manufacturing, the scheduling of maintenance operations on the machines used in manufacturing is a critical function, since:

- The maintenance operations can be expensive to perform, so we only want to perform them when necessary.
- If maintenance is delayed too long, machines may run sub-optimally or break down (thus requiring even more expensive unplanned (corrective) maintenance).
- A machine that is undergoing maintenance may be partly or wholly unavailable for production operations.

Maintenance scheduling addresses the issue of how to schedule maintenance operations over a period of time in a manufacturing plant order to maximize the up time of machines and minimize disruption to production operations.

We have developed an optimization tool to generate maintenance schedules for IBMs East Fishkill, New York 300mm semiconductor manufacturing plant (Yario 2005) for a two week horizon. The tool determines when maintenance operations should take place on which machines, subject to the availability of technicians who carry out the maintenance operations. The goals of the optimization tool are to minimize disruption to the production operations in the plant caused by maintenance operations and to levelize the use of maintenance technicians (who are required to perform maintenance operations) over time.

In the sections which follow, we give a description and problem formulation of the maintenance scheduling problem in semi-conductor manufacturing and discuss some of the challenges in solving this problem. We present a hybrid constraint programming and mixed-integer programming solution approach which we have used in an operational system that is now in use within IBM.

Problem description

Application background

A semiconductor processing fab is responsible for the front-end of the chip manufacturing process, in which a series of processing steps are performed, layer by layer, on the surface of cylindrical silicon wafers inside ultra-clean rooms, in order to produce sets of custom integrated circuit devices on each wafer. The operation of the semiconductor processing fab can be characterized by a high degree of complexity with respect to manufacturing operations and process alternatives, with respect to the optimal allocation of constrained resources in a dynamic production environment, and with respect to the massive scale of data that is collected from the individual process tools and wafer test devices. A semiconductor wafer fabrication factory is a complex manufacturing environment which may consist of hundreds of product routes, thousands of process steps with re-entrant ows through hundreds of tools. Stochasticity is introduced by the inherent variability in processing time, testing time, unplanned tool outages and wafer re-work.

Inputs

In the semi-conductor manufacturing environment there are number of different types of maintenance operations to consider:

- Preventative maintenance operations: periodic maintenance recommended by manufacturer of machine, to be carried out at regular time intervals (for example, once every six months).

- Trigger maintenance operations: required after a machine reaches a certain state. For example, a wafer count trigger is reached after a certain number of wafers have been processed on a machine.
- Unplanned maintenance operations: unforeseen machine breakdowns which require the machine to be taken down in order to repair.

The purpose of maintenance scheduling is to generate a detailed schedule for preventative and trigger maintenance operations[1]. For each maintenance operation, we are given a release date, a due date, a processing time, a machine or machine part (on which maintenance is to be performed) and a demand for some constant number of technicians throughout its entire processing time to perform the maintenance. Note that maintenance may sometimes be performed on only one part of a machine (for instance a single chamber of a lithography machine) or on an entire machine. When a maintenance operation is performed on an entire machine, no other maintenance operations can be performed at the same time on the machine, nor is the machine be available for the processing of production operations. When a maintenance operation is performed on a part of a machine, other maintenance operations can be carried out in parallel on other parts of the machine.

The machines in a semi-conductor fab can be partitioned into a number of *toolsets*. A toolset is a set of machines of a certain type (e.g. lithography) manufactured by a certain vendor. Maintenance operations need to be performed by maintenance technicians who are certified to work on machines belonging to a particular toolset. In practice, we have observed that maintenance technicians are mostly certified to work on machines belonging only to a single toolset: in rare cases a technician might be certified to work on machines belonging to two toolsets. We exploit this property in our solution design. For each toolset, we are given a timetable specifying the number of maintenance technicians available during each time period (shift) for performing maintenance operations. The available "capacity" of maintenance technicians for each toolset is the bottleneck resource, limiting how many maintenance operations can be performed when generating a maintenance schedule[2].

To summarize, we consider the following inputs to the maintenance scheduling problem for semi-conductor manufacturing:

- A set of tools and toolsets: each tool belongs to a unique toolset.
- A timetable of available internal technician capacity over time for each toolset within the plant. Technicians are qualified to perform maintenance operations on one (and sometimes two) toolsets. Each maintenance operation is performed on a specific tool, and (usually) cannot be interrupted once processing has begun. During the processing of a maintenance operation, the tool is unavailable for other operations (e.g. production operations).
- A set of maintenance operations to be scheduled, where each operation is associated with a release date, a due date, a processing time, a tool, a toolset and a demand for a fixed number of technicians during the processing time of the operation.

[1]Currently we do not consider unplanned maintenance at all. We are considering taking into account statistics concerning unplanned maintenance (such as mean time between failures) in order to reserve maintenance technician capacity when likelihood of unplanned maintenance is high.

[2]In practice, if we have to schedule operations such that they exceed the available number of maintenance technicians needed to perform them, we can sub-contract some operations to "external" technicians (from the machine vendor). This is very expensive however, and should be avoided.

Objectives

In practice feasible schedules satisfying release dates, due dates and capacity constraints on maintenance technician capacity are usually easy to generate: resource contention is not the main challenge in generating good maintenance schedules. Instead, the real challenge is managing the multiple, complex objectives and user preferences.

Resource leveling The first objective that we consider relates to the utilization of the available maintenance technicians for a toolset over time. When the use of maintenance technicians is spread out over time, it is more likely that some surplus technicians will be available to carry out any unplanned maintenance that may occur (illustrated in Figure 1(right)). In time periods when many technicians are busy, it is less likely there will be surplus technicians to deal with any unplanned maintenance that may occur during the busy period (illustrated in Figure 1(left)). Unplanned maintenance can be very expensive and disruptive to production operations, so in general it is preferred that we "levelize" the use of maintenance technicians over time (as illustrated in Figure 1(right)) so that some technicians are always available to handle unplanned maintenance should it become necessary. We formulate this requirement as an objective that we *minimize the number of technicians* that we utilize throughout the entire schedule in order to perform all of the pending maintenance operations (alternatively we minimize the *maximum* number of technicians that are active (performing maintenance) at any one time period)).

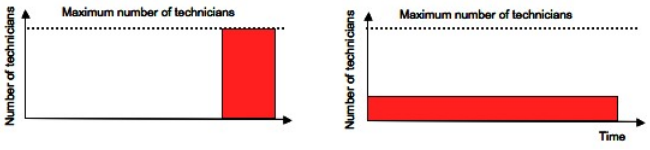

Figure 1: Bad (left) and good (right) maintenance technician utilization

Minimizing disruption The second objective relates to minimizing the disruption to production that occurs as a result of taking machines out of service in order to perform maintenance on them. Typically, machines in a semiconductor manufacturing fab process lots consisting of a number of silicon wafers. At any one time when a machine

is busy processing a lot, there may be a number of wafers waiting in a queue to be processed by the machine. This number of wafers is the Work In Process (WIP), which is specified for a machine and a time period. When maintenance is performed on a machine, all production is stopped on that machine. As well as delaying wafers that need to be processed on the machine, this can lead to starvation of downstream processes for the wafers, resulting in machine under-utilization. Ideally, we would like to minimize such disruption by performing maintenance operations on tools during time periods when there is no WIP at all, or as little WIP as possible. Figure 2 illustrates a maintenance schedule for a set of tools belonging to a toolset, where the WIP levels for each tool and time period are plotted in the background.

We formulate the objective for minimizing disruption in the following way: given w_{mt} which denotes the level of WIP on machine m in time period t, and a maintenance operation k on machine m which starts in time period s and finishes in time period e, the WIP disruption of this operation is $\sum_{t\in[s,e]} w_{mt}$. We wish to minimize the total WIP disruption for all scheduled maintenance operations.

The difficulty we face with using this objective in a scheduling context is that typically (and certainly for East Fishkill) we do not know what the WIP levels will be in the fab for each tool over the scheduling horizon. Operations in semiconductor fab are usually very dynamic, as wafers can make multiple passes through various processes based on the results of tests of the effectiveness of each production step. Uncertainty also arises due to unplanned machine breakdowns. Detailed production scheduling is usually done using dispatch rules applied whenever a machine becomes available for processing. As such, there is no longer term production schedule we can refer to in order to determine what the WIP levels will be for each tool that we could use in a formulation of the maintenance scheduling problem. Instead we determine the Expected Work In Process will be for each tool and time period, based on a simulation of the flow of wafers through the fab. For this we use the IBM Research WIP Simulator, developed specifically for semi-conductor manufacturing (described in (Bagchi et al. 2008)). We run 20 replications of the WIP simulator for a whole scheduling horizon, based on a division of the horizon into one hour time buckets. From the output of WIP simulator, we obtain the expected WIP level for each machine (tool) during each one hour time bucket. Note that the results of the WIP simulator are usually quite accurate in the short term (1-3 days) and much less accurate in the longer term (up to two weeks). As a result, maintenance scheduling is usually performed every day. Some rescheduling of operations may occur over a number of days and runs of scheduler, taking into account revised expected WIP estimations.

Minimizing earliness and tardiness The third objective relates to minimizing the long term costs of performing periodic maintenance. There is some flexibility in determining when a maintenance operation j can be performed in the schedule, specified by it's release date and it's due date. However for periodic maintenance, the interval of time that can elapse between the completion time e_i of one operation i and the start time s_j of the following operation j (where each operation performs the same periodic maintenance) on a tool should not exceed a given period duration $D_{i,j}$[3]. This gives rise to earliness and tardiness costs for scheduling maintenance. Tardiness costs result from scheduling an operation j to start at some time s_j such that $s_j > e_i + D_{i,j}$. Earliness costs result from scheduling an operation j to start at some time s_j such that $s_j < e_i + D_{i,j}$. The due date d_j of an operation j is calculated such $d_j = e_i + D_{i,j}$, i.e. scheduling an operation at it's due date incurs no earliness or tardiness cost.

We are given an earliness penalty α_j and a tardiness penalty β_j for each operation j. Given the completion time C of operation j in a feasible schedule, the earliness / tardiness cost et_{jC} for this operation can be computed as $et_{jC} = \max(\alpha_j(d_j - C), \beta_j(C - d_j))$.

Mathematical formulation

We model the maintenance scheduling problem using a time-indexed integer programming formulation. We first present the notation for the problem parameters and then introduce our basic mathematical model. In the case of the IBM East Fishkill plant, most or all of the technicians are certified to perform maintenance only on tools in a single toolset. As a result, there are no constraints in the model that span different toolsets. Therefore we can decompose the full problem of scheduling all maintenance operations in the fab into a set of disjoint maintenance scheduling problems, where each sub-problem schedules the operations for a single toolset.

Table 1: Notation

T:	the time horizon of the schedule, indexed from $[1, T]$
O:	the set of all maintenance operations
r_i:	the release date of operation i
d_i:	the due date of operation i
p_i:	the processing time of operation i
q_i:	the number of technicians needed to perform operation i
M:	the set of all tools needing maintenance
m_i:	the tool on which operation i performs maintainance
$J(m)$:	the set of maintenance operations performed on tool m
C:	the maximum number of technicians available
c_t:	the number of technicians available at time t
α_i:	the per time unit earliness penalty for operation i
β_i:	the per time unit tardiness penalty for operation i
et_{it}:	the earliness/tardiness cost of operation i in period t
w_{mt}:	the WIP on tool m in period t

The binary decision variables of the formulation are:

$$x_{it} = \begin{cases} 1, & \text{if operation } i \in O \text{ starts in period } t \\ 0, & \text{otherwise} \end{cases}.$$

where $t \in \{r_i, \ldots, T - p_i\}$ for all $i \in O$. Using these time-

[3]This duration is specified by the manufacturer of the tool.

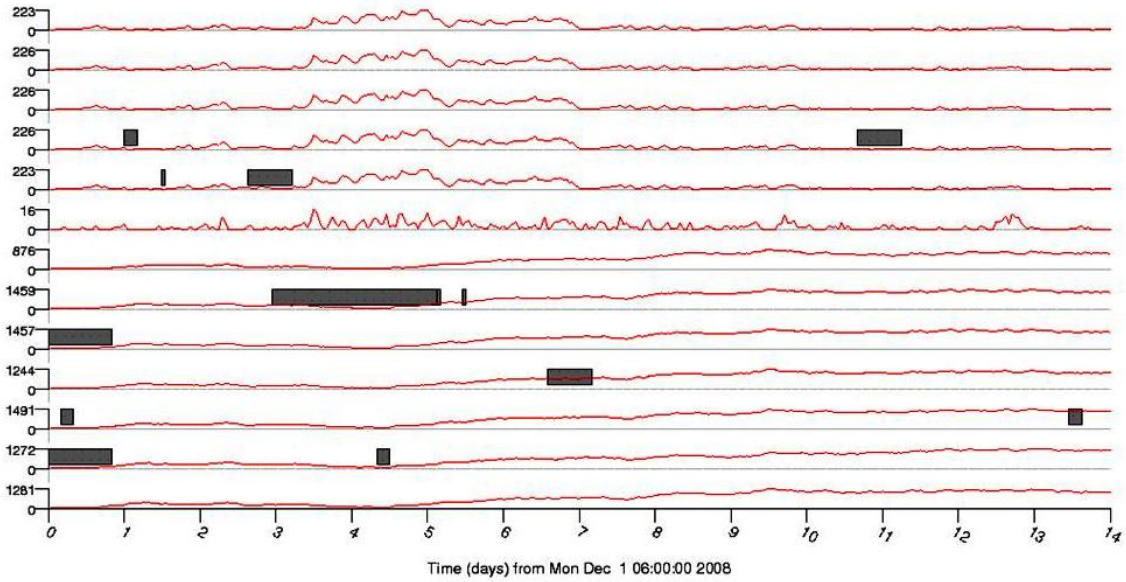

Figure 2: Illustration of a maintenance schedule for a single toolset. Each row presents a Gantt chart representation of a schedule for a single tool in the toolset. The plotted line represents the projected work in progress (y-axis) for the tool over each time period (x-axis) in the schedule. Grey boxes represent the start time and duration of scheduled maintenance operations on each tool.

indexed decision variables, the formulation becomes:

$$\min\ \lambda_1(C-z) + \lambda_2 \sum_{i\in O}\sum_{s=r_i}^{T-p_i}\sum_{t=s}^{s+p_i} w_{m_i t} x_{is} + \lambda_3 \sum_{i\in O}\sum_{t=r_i}^{T-p_i} et_{it} x_{it} \tag{1}$$

$$\sum_{t=r_i}^{T-p_i} x_{it} = 1 \qquad \forall i \in O \tag{2}$$

$$\sum_{i\in J(m)}\sum_{s=\max\{t-p_i,r_i\}}^{\min\{t,T-p_i\}} x_{is} \le 1\ \ \forall t \in [1,T],\ \ \forall m \in M \tag{3}$$

$$\sum_{i\in O}\sum_{s=\max\{t-p_i,r_i\}}^{\min\{t,T-p_i\}} q_i x_{is} + z \le c_t\ \ \forall t \in [1,T] \tag{4}$$

$$x_{it} \in \{0,1\} \qquad \forall i \in O\ \ \forall t \in [r_i, T-p_i] \tag{5}$$

$$z \in \{0,\ldots,C\} \tag{6}$$

The objective (1) represents a weighted sum of the objectives. The first term of the objective maximizes the number of available maintenance technicians in each period that are not used in the schedule (and hence are avialable for unplanned maintenance). The second term minimizes disruption to production operations and the third term minimizes earliness and tardiness costs. Constraints (2) state that each maintenance operation can be started only once. Constraints (3) are resource constraints stating that at any given time on each tool at most one operation can be processed. Constraints (4) are resource constraints stating that at any given time we cannot exceed the available maintenance technician capacity.

Side constraints

In addition to the core problem formulation we are given a number of side constraints for the problem arising from user preferences on the form of the schedule. These user preferences are not considered to be hard constraints, but we should satisfy them if possible regardless of their impact to the problem objective. We briefly discuss two of the side constraints in this section.

Follow-up maintenance constraints When a technician has completed a maintenance operation on a particular machine, there is a likelihood that the machine will require some follow-up maintenance for up to six hours afterwards. Ideally, this follow-up maintenance should be performed by the same technician who performed the original maintenance operation i.e. we wish to reserve the technician who performed the operation to prevent him/her from performing any further maintenance operations anywhere else in the schedule for six hours from the completion of the operation. We wish to maximize the number of operations for which we can satisfy this preference.

Separation constraints Given two maintenance operations i and j to be performed on the same machine, we would prefer to schedule them in one of the following ways in decreasing order of preference:

1. i and j are separated by at least 24 hours;
2. i and j are separated by at least 12 hours;

3. i and j are scheduled continuously, so there is no gap between the end time of i and the start time of j or vice-versa.

If we assign weights w_i to each of the preferences (1)-(3) presented above, we wish to maximize the weighted sum of the satisfied preferences for all of the maintenance operations that are scheduled in the solution.

Solution approach

The solution approach we developed is motivated by the following observations, based on typical problem data:

1. Generating a feasible maintenance schedule is very easy: the main resource bottleneck is the availability of maintenance technicians.
2. Generating an optimal maintenance schedule is difficult, due to the multiple, irregular objectives and preferences.

Scheduling problems in manufacturing have been solved using a variety of techniques from local search, genetic algorithms and dispatch rules to exact methods based on branch and bound search, constraint programming and mixed integer programming. Since we wish to find an optimal solution, we focus on using exact methods. Constraint programming (Baptiste, Pape, and Nuijten 2001) and mixed integer programming with time-indexed formulations are often the solution techniques of choice for modelling scheduling problems with complex objectives and side constraints. These techniques have different strengths and weaknesses:

1. Constraint programming solvers can model scheduling problems compactly using an event-based formulation, and can be very successful at finding good feasible solutions to problems which are highly resource-constrained. Constraint programming is not always suitable for solving problems with complex, non-convex objectives [4].
2. Mixed-integer programming solvers using time-indexed formulations are able to model scheduling problems with complex, non-regular objectives[5]. However the formulations can be very large, since the number of decision variables is dependent on the length of the time horizon. As such, this approach is often limited to small problems[6].

We rank the objectives and preferences, considering them in the following order of decreasing importance:

1. resource levelling: minimize the total number of maintenance technicians required in the schedule.
2. separation constraints: maximize the number of consecutive pairs of operations on the same machine satisying the separation constraints.
3. follow-up constraints: maximize the number of operations which have a technician available for follow-up maintenance.
4. disruption: minimize the overlap of maintenance operations with work in progress.
5. earliness-tardiness costs.

We use a solution approach inspired by lexicographic goal programming. In lexicographic goal programming, we first solve the problem with respect to the most important objective only (we ignore all other objectives). Let f_1 denote the objective value for the first objective in the solution to this problem. We now add a new *constraint* to the problem model, stating that the value for the first objective must be equal to f_1. We then solve the problem for the second objective only, but with the first objective now represented as a constraint in the model. Subsequently, we add a second constraint to the model based on the objective value found for the second objective. We continue this process until we have solved the problem for all objectives.

The objectives for resource levelling, separation constraints and follow-up constraints can all be solved very efficiently using constraint programming [7]. Since finding a feasible solution to the maintenance scheduling problem is very easy using constraint programming (taking less than 0.1 second), we can determine the minimum number of technicians required by solving a series of feasibility problems, where for each problem we set the number of technicians available to a fixed value. We use binary search on the number of technicians to determine the smallest number for which we can find a feasible solution that schedules all the pending maintenance operations. A similar approach can be used to determine the best objective value for the separation and follow-up constraints.

We solve for the objectives concerning disruption and earliness-tardiness using mixed-integer programming. We use the time-indexed formulation with some additional cuts. In practice, the solution times to solve the mixed integer programming model are much slower than that for the constraint programming solver, since the formulation can be quite large (a single toolset may have up to 100 operations to be scheduled over a two week time horizon). We discretized time into 15 minute time buckets, giving us solution times for the MIP solver on the order of 5-20 minutes (using CPLEX version 11).

[4] ILOG's CP Optimizer (Laborie et al. 2008) has a flexible representation for modelling objective functions, however it requires them to be in a semi-convex form.

[5] Time-indexed formulations have been found to give good linear programming relaxations when the objective function is of the form $\sum_j f_j(C_j)$ (where C_j is the completion time of job j), even if f_j is not non-decreasing (non-regular).

[6] There are specialized branch and bound techniques based on linear programming (or shortest path) relaxations and Lagrangean relaxation or column generation (e.g. (Sourd 2009)). We do not consider them here since we wish to be flexible with respect to accomodating side constraints to the scheduling model.

[7] Space prevents us from discussing details of the constraint programming solver used in this paper. The solver has been developed internally at IBM Research, and uses depth-first chronological backtracking, the SetTimes branching heuristic and the timetable resource constraint propagator (Baptiste, Pape, and Nuijten 2001) (stronger propagation than timetable was not found to be necessary).

Summary

We have present a maintenance scheduling problem for a semi-conductor fabrication plant. We have developed a goal programming approach combining constraint programming and mixed-integer programming which exploits the strengths of both solution techniques. The scheduling system we have developed based on this solution approach has been deployed at IBM's 300mm East Fishkill semiconductor fab.

References

Bagchi, S.; Chen-Ritzo, C.; Shikalgar, S.; and Toner, M. 2008. A full-factory simulator as a daily decision-support tool for 300mm wafer fabrication productivity. In Mason, S. J.; Hill, R. R.; Moench, L.; and Rose, O., eds., *Proceedings of the 2008 Winter Simulation Conference*.

Baptiste, P.; Pape, C. L.; and Nuijten, W. 2001. *Constraint-Based Scheduling - Applying Constraint Programming to Scheduling Problems*. International Series in Operations Research and Management Science. Springer.

Laborie, P.; Rogerie, J.; Shaw, P.; Vilim, P.; and Wagner, F. 2008. ILOG CP optimizer: Detailed scheduling model and OPL formulation. Technical Report ILOG Technical report number 08-002, ILOG, `http://www2.ilog.com/techreports`.

Sourd, F. 2009. New exact algorithms for one-machine earliness-tardiness scheduling. *INFORMS Journal of Computing* 21:167–175.

Yario, J. 2005. 2005 top fab: IBM. In *Semiconductor International*.