IBM Solves a Mixed-Integer Program to Optimize Its Semiconductor Supply Chain

Article *in* Interfaces · October 2006

DOI: 10.1287/inte.1060.0238 · Source: DBLP

3 authors, including:

Brian T Denton
University of Michigan
108 PUBLICATIONS **4,726** CITATIONS

SEE PROFILE

R. John Milne
Clarkson University
59 PUBLICATIONS **1,076** CITATIONS

SEE PROFILE

Interfaces
Vol. 36, No. 5, September–October 2006, pp. 386–399
ISSN 0092-2102 | EISSN 1526-551X | 06 | 3605 | 0386

informs

DOI 10.1287/inte.1060.0238

IBM Solves a Mixed-Integer Program to Optimize Its Semiconductor Supply Chain

Brian T. Denton
Division of Health Care Policy and Research, Mayo Clinic, Pavilion Building 3-06, Rochester, Minnesota 55906, denton.brian@mayo.edu

John Forrest
IBM T. J. Watson Research Center, PO Box 218, Yorktown Heights, New York 10598, jjforre@us.ibm.com

R. John Milne
IBM Systems and Technology Group, 1000 River Road, Essex Junction, Vermont 05452, jmilne@us.ibm.com

IBM Systems and Technology Group uses operations research models and methods extensively for solving large-scale supply chain optimization (SCO) problems for planning its extended enterprise semiconductor supply chain. The large-scale nature of these problems necessitates the use of computationally efficient solution methods. However, the complexity of the models makes developing robust solution methods a challenge. We developed a mixed-integer programming (MIP) model and supporting heuristics for optimizing IBM's semiconductor supply chain. We designed three heuristics, driven by practical applications, for capturing the discrete aspects of the MIP. We leverage the model structure to overcome computational hurdles resulting from the large-scale problem. IBM uses the model and method daily for operational and strategic planning decisions and has saved substantial costs.

Key words: computers: computer sciences; programming: integer, applications.

Semiconductor manufacturing, the epitome of high-technology manufacturing, requires substantial capital investments, often measured in billions of dollars. Because of high investment costs for facilities, semiconductor manufacturers often face the possibility of limited resources and multiple bottlenecks throughout the supply chain. To cope with limited capacity and long manufacturing lead times, they must allocate assets to customer demand to achieve optimal production plans, coordinate interplant logistics, and high-quality customer service. Supply chain optimization (SCO) is a key to profitability in the semiconductor industry. Systems that support SCO are considered business-critical and must incorporate a broad range of planning criteria and constraints. IBM Systems and Technology Group uses SCO models to solve large-scale supply chain planning problems (Lyon et al. 2001).

Firms use SCO models to control the activities of an extended enterprise, such as sourcing among multiple plants, interplant shipping logistics, and developing production plans for manufacturing plants within the enterprise. The time horizons for such planning models range from several months to many years, depending on the decisions they are intended to support. Their uses include controlling daily production schedules and coordinating targets that factory execution systems must meet. For some applications, a horizon of several months, composed of daily planning periods, may be appropriate. For others, where long-range strategic decisions are called for (whether to build a new facility or make a major investment in capacity), a time frame of five or more years, with weekly or monthly planning periods, may be appropriate. Whatever the types of application, SCO models must take into account a range of criteria if they are to be useful.

Complex supply chains are commonly modeled as linear programs (LPs), which can effectively trade off a broad range of criteria. However, to model semiconductor supply chains accurately, one must include discrete aspects of decision making, which requires solving a mixed-integer program (MIP). The corresponding MIP is NP-complete. Therefore, general

methods of solution (for example, branch-and-bound, branch-and-cut) are practical only for fairly small problems or special classes of problems. For large-scale problems, one must take advantage of properties of the problem structure. Furthermore, because computing optimal solutions is not possible, the focus is on developing fast heuristics that can find good feasible solutions in reasonable computation time. Developing appropriate heuristics is especially difficult in a production environment; such an environment operates according to rigorous scheduling requirements and demands reasonable and reliable computation times.

We designed, developed, and deployed several heuristic methods for solving an SCO model as part of an advanced planning system (APS) for the centralized planning of IBM's Systems and Technology Group's semiconductor supply chain. Here we describe some important developments in supply chain optimization at IBM since the 2000 Edelman competition, in particular, the development of a MIP model and solution methodology.

Semiconductor Industry Background

Semiconductor manufacturing involves a range of activities, including everything from growing silicon ingots (the source of silicon wafers upon which integrated circuits are grown) to the actual placement and soldering of finished chips to a printed circuit board. Initially, raw wafers, cut from a silicon ingot, are processed through a specific sequence of work centers. The goal of wafer fabrication is to build a set of devices (integrated circuits) on the surface of the silicon wafer according to a specified circuit design. At a high level, this process consists of repetitions of four essential steps: deposition, photolithography, etching, and ion implantation. The first three steps are the means by which material is deposited on the wafer surface (deposition), patterned by protecting parts of the surface that correspond to circuit structures (photolithography), and processed to remove the unprotected material (etching). The last step, ion implantation, is the means by which the conductive properties of the silicon wafer are modified, which is a key to building circuit components, such as transistors. Together these steps form the manufacturing process for patterning the surface of the wafer with materials that have special dielectric properties (conductors, insulators) according to precise circuit-design specifications. The steps are repeated many times to build up a sequence of layers; the layers correspond to building circuit components (diodes, resistors, transistors), and subsequent layers correspond to building metal interconnections between the circuit components. The finished devices are three-dimensional structures built on the two-dimensional surface of the wafer. A single device may have millions of circuit elements with line widths measured in nanometers.

Once devices have been built on a wafer, they are tested and their quality attributes (speed, power consumption) are recorded for later reference. Wafers are then diced and sorted into individual devices and subsequently bonded to a substrate and packaged to assemble a module. The modules, which are further tested to determine electromagnetic and thermal characteristics, are eventually assembled onto printed circuit boards to make cards. Finally, the cards are tested, and those that pass inspection are eventually used to assemble a wide range of finished electronic products (for example, servers, cell phones, and CD players). From the point of view of semiconductor manufacturing, the modules and cards are, by and large, the finished products taken to market. However, depending on the particular markets the firm is engaged in, there may also be demand for finished wafers.

Modern wafer-fabrication facilities are designed as large clean rooms with central corridors lined with manufacturing bays on both sides: photolithography bays, diffusion bays, deposition bays, plasma etching bays, and bays for other critical manufacturing operations. Wafers are manufactured in discrete lots, which travel between bays in sealed wafer carriers. Because of the tool setups needed for processing lots in various operations, wafer carriers are normally required to be full. Thus, production starts must be in integer multiples of the carrier size. Within certain manufacturing bays, there are batch constraints that determine the number of wafer lots that can be processed at a time. For example, a diffusion bay typically contains several vertical furnaces that are used at various stages in the manufacturing process (for example, in annealing). These furnaces can typically hold 100 to 200 wafers at a time, and planners

impose a minimum production start quantity to avoid low utilization of furnaces. These container-size and batch-size constraints are lot-size constraints.

To create manageable SCO models, we introduce some abstraction from the complex routing of lots through the production process. We project the continuous process onto a discrete set of part numbers (PNs). We use a bill of material (BOM) that specifies the components of each PN to generate a graph representation of the components needed for finished products (Figure 1). The fabrication process starts with raw wafers, which are processed into front-end-of-the-line (FEOL) wafers, which are wafers with semifinished devices that have circuit components not yet connected by metal leads. The FEOL wafers are then used to build back-end-of-the-line (BEOL) wafers in which the circuits are completed by introducing several layers of metal interconnections. After further processing, the devices are tested and sorted (or binned) into categories, such as fast, medium, and slow. The distribution of attributes results from random variation in the manufacturing process. At this stage, high-grade devices (for example, faster) can be substituted for low-grade devices (for example, medium or slower).

The facility packages devices to assemble modules, which may include multiple devices (multichip modules), and there may be multiple assembly processes that could be used within a given packaging facility. The finished modules are tested to determine whether the devices have successfully negotiated the

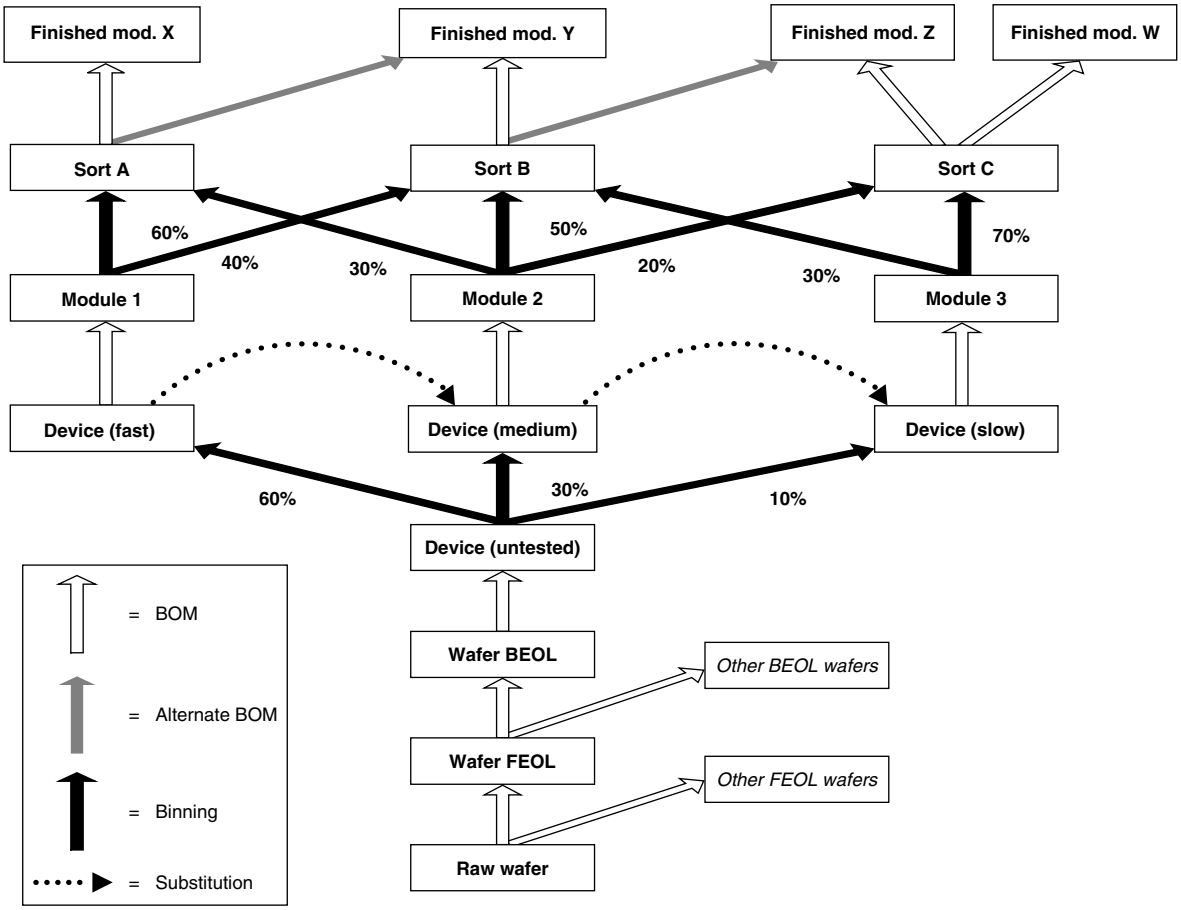

Figure 1: For semiconductor production, material flows throughout the supply chain from raw silicon wafers (bottom) to finished modules (top). For a given module—X, Y, Z, or W—there may be multiple assembly processes; for example, module Y could be built using Sort A or Sort B.

packaging process. This results in a second round of binning. Final processing involves capping modules and assigning customer PNs and other relevant packaging information. For a given module, there may be multiple assembly processes (Figure 1). Two building processes may be differentiated by processing costs and may require different capacity resources. Further assembly may be carried out to create printed circuit boards.

In general, large semiconductor manufacturing firms have many facilities and outsource some operations. IBM has multiple semiconductor fabrication facilities, test facilities, and packaging facilities. The SCO model must link the optimization of manufacturing plans at individual sites via interplant logistics that describe the flow of material among facilities. It must also consider such factors as capacity allocation and utilization of work-in-process (WIP) inventory. Furthermore, because multiple sites in the supply chain may perform overlapping functions, the model must balance the use of resources at the various sites.

Planners of semiconductor supply chains have traditionally relied on software systems based on OR/MS methods. For instance, at individual manufacturing sites, they often use material-requirements-planning (MRP) systems to develop uncapacitated plans for meeting the sites' demands, demand-management (DM) tools to manage forecasts of enterprisewide demand at different hierarchical levels within the organization, and available-to-promise (ATP) applications to commit supply to orders as they are booked in real time. More advanced software tools introduce capacitation into the planning process and are based on heuristics for rationing finite capacity or formal mathematical programming models for determining optimal solutions. IBM Systems and Technology Group has developed a complete suite of these tools.

Related Literature

Researchers have explored several supply chain planning problems, including some directly related to our modeling and methodology. Uzsoy et al. (1992) explored production planning, and Uzsoy et al. (1994) surveyed shop-floor scheduling in the semiconductor industry. They focused on studies of a single plant rather than central planning of an extended enterprise. Tayur et al. (1998) reviewed the general supply chain literature. Arntzen et al. (1995) described applications of supply chain management at Digital Equipment Corporation. Leachman et al. (1996) described a comprehensive system for production planning at Harris Semiconductor Corporation. Camm et al. (1997) discussed supply chain restructuring decisions in designing product sourcing and distribution systems. Bermon and Hood (1999) described a model for planning capacity at a semiconductor plant. Lin et al. (2000) described a case study of supply chain planning in IBM's Personal Systems Group.

Lot sizing is a particularly important issue in planning semiconductor supply chains. Raw wafers, the initial material released into a wafer-fabrication facility, are issued in discrete lot sizes (typically 25 wafers per lot). Furthermore, there are many batch tools on the fab floor; batch furnaces, for example, may process between 100 and 200 wafers at a time. The introduction of lot-size constraints results in a MIP in which production starts are constrained to be at least a certain minimum quantity and must be released in discrete multiples of the allowable lot size. Treatment of lot sizes as decision variables, that is, the a priori determination of appropriate lot sizes, has attracted much attention. For the simple single-product problem, Wagner and Whitin (1958) developed a dynamic programming method. Kuik et al. (1994) provided detailed survey of lot-sizing models. More practical multiproduct problems have been studied recently by Belvaux and Wolsey (2000), who described a branch-and-cut approach based on combining several types of cuts. Belvaux and Wolsey (2001) described methods for reformulating lot-sizing problems to achieve substantially improved computation times. All of these authors treated the discrete decision of whether or not to produce in a given time period. However, none of them considered our situation, in which a production release in a given time period must be a multiple of a predetermined lot size.

Demand prioritization is another important aspect of supply chain planning. In general, LP models are not well suited to preemptive demand prioritization. Leachman et al. (1996) developed a method for sequentially considering demand classes, one at a time, with constraints added successively to guarantee

that lower-priority classes do not steal from higher-priority ones. We developed a multistage algorithm to handle preemption for multiple demand classes at a time within our model. Our approach provides solutions that reflect preemptive prioritiation, and it is computationally efficient.

The Mixed-Integer Programming Model

In 1995, IBM decided to enter the OEM market and move away from being a captive supplier of microelectronics products. As part of this plan, it centralized supply chain planning. In support of the plan, our team at IBM developed OR-based software solutions, including a central planning engine (CPE) to balance the supply chain's resources against a central statement of semiconductor demand. We designed the CPE to balance the need to handle extremely large models against the need for high-quality solutions in a timely manner by simultaneously using fast rules-based heuristics for standard material flows and a MIP model for complex material flows. The CPE automatically partitions the semiconductor supply chain into those sections that may be solved well with heuristics and sections better suited to formal mathematical programming (Hegde et al. 2004).

The MIP model is formulated as a cost-minimization problem subject to a set of constraints describing material flows and other aspects of the supply chain (appendix). The objective function includes several decision variables and associated cost penalties, with provisions for material substitutions, interplant shipments, customer shipments, inventory holding, and variable costs associated with production starts. The model includes both soft and hard capacity constraints; the soft constraints may be violated subject to a cost penalty, whereas the hard constraints must be satisfied. There are variables in the objective function that are related to penalties for violating the soft constraints. The soft capacity constraints relate to model situations in which nominal planned capacity may be increased at a cost (for example, by outsourcing, or short-term leasing of tools). We assume that demand that cannot be satisfied on time is back-ordered, and costs are accrued in each planning period in which that takes place. The core set of model constraints includes back-order-balance constraints, inventory-balance constraints, capacity constraints, sourcing constraints, and discrete lot-size constraints. The decision variables include $P =$ production starts, $I =$ inventory, $F =$ shipments, $T =$ internal shipments, and $L =$ material substitutions.

Constraints (i) of the MIP model represent back-order balancing from one planning period to the next, given the shipment decision variables, F_{tmakq}, and new demand arising in the period. If demand in a planning period for a particular PN, customer location, and demand class exceeds the quantity shipped, then the excess results in an increased back order and an associated cost. Back orders accumulate over time until sufficient shipments can be made to cover them. There may be multiple locations that can ship to cover a given demand.

Constraints (ii) of the MIP model represent the inventory balance of material flows of PNs between stocking locations in the supply chain as well as through planning periods in the planning horizon. These constraints form the backbone of the network of stocking points. They keep track of the movements of assets via interplant shipments and the movements of WIP between stocking points within a plant as a result of production. They also reflect the movement of material along multiple production paths (Figure 1). Flows of production starts from one stocking point to the next are governed by yield losses, Y_{tmae}, which define the quantity of material coming to stock, and associated cycle times, ct_{tmae}, which define the time from production start to production stock. The available paths represented by assembly component relationships within a plant are defined via BOM information; they include multiple processes for building the same PN and the binning of PNs into multiple PNs (for example, fast, medium, and slow). The inventory-balance constraints also define the feasible flow of materials based on allowable material substitutions.

Constraints (iii) of the MIP model represent restrictions on capacity that limit production starts for particular PNs at a given plant in each planning period, based on available work-center capacity for the various resources in the supply chain. These constraints may relate to any type of finite capacity that must be allocated to create feasible production plans and may include tools for wafer fabrication, testing, and assembly. In general, mapping PNs to capacity resources

may involve complex many-to-many relationships that couple otherwise distinct product types. Planners can adjust the constraints to also capture complexities pertaining to the timing of capacity utilization. For instance, they can use time offsets in which a particular PN uses capacity at some future time after a production start. Also, in some cases in which there are secondary opportunities (for example, outsourcing), capacity constraints may be defined with a slack variable (soft constraint) that allows for violation of a constraint subject to a cost penalty in the objective function.

Constraints (iv) of the MIP model, the sourcing constraints, provide balance across the supply chain for those items that can be produced at multiple locations or procured from multiple suppliers. For instance, multiple wafer-fabrication facilities may produce the same devices, or multiple test facilities may test parts. In such cases, planners must decide on sources based on strategic objectives instead of breaking ties arbitrarily. These soft constraints, with associated slack variables S_{tzau} and G_{tzau}, indicate target shipping quantities to be supplied by sourcing locations (for example, a 70 to 30 percent split between two locations). Planners use the constraints to apply sourcing rules within the supply chain (internal shipments) as well as shipment of finished goods to customers.

Constraints (v) of the MIP model are discrete lot-size constraints that require that production starts correspond to rules governing allowable start quantities based on user-defined parameters. These rules define the minimum and multiple quantities allowed in production releases. The minimum quantities may correspond to policy-based constraints that limit the costs associated with setups, whereas the multiple quantities impose the requirement that production starts be in multiples of container size. For instance, because wafers are processed in carriers that hold 25 wafers, a rule that requires a minimum of 100 wafer starts would have allowable quantities in the set $\{100 + 25n, n = 0, 1, 2, \ldots\}$.

Solution Methodology and Implementation

IBM Systems and Technology Group has a long history of using advanced planning systems, including MRP and heuristic-based capacitated planning tools. Since 2000, the semiconductor CPE has undergone continuous improvement, with many functional enhancements that capture new aspects of the business environment as well as substantial performance improvements for solving large-scale SCO models. We implemented our new MIP-based functionality in a controlled fashion, asking users to test new features systematically, to provide feedback for improvements, and subsequently to approve the new functions for deployment in production. We will describe three examples of heuristic methods that were developed to facilitate the real-world application of the MIP model.

LP Presolve Heuristic

The problems encountered in practice may have millions of variables and constraints. Therefore, solving even the LP relaxation of the MIP can be challenging. The group's first implementation of the model was an LP that did not include discrete lot-sizing constraints. We initially applied the model to small sections of the supply chain that were particularly difficult to schedule with rules-based heuristics. This initial implementation resulted in problem sizes for which generic application of a commercial LP solver was sufficient. However, over time as we added new constraints to enrich the model and additional sections of the supply chain, the computation times became unacceptably long.

To solve the very large-scale problems that now arise in practice, we must take advantage of the problem structure. In semiconductor supply chain models, different product groups often share similar capacities and hence constraints; similarly, multisourcing constraints link production across multiple facilities. Therefore, we typically cannot readily decompose the LP model. By relaxing some of the linking constraints, we can create a decomposable problem and then apply methods like Dantzig and Wolfe's (1960) decomposition. However, in experimenting with such methods for these types of LP models, we found that they are typically slow to converge. One of the approaches we have developed for solving large-scale models employs a decomposition-based heuristic as a presolve stage to get a near-optimal solution before obtaining a final one. This method exploits the supply chain structure of the MIP model.

Our method involves first acquiring a feasible solution. A feasible solution can be obtained by setting to zero all variables except (a) the back-order variables, B_{tmkq} (which we set assuming that all demand is back-ordered in each period) and (b) the slack variables for soft constraints, such as S_{tzau} and G_{tzau}. This allows fast generation of a feasible solution. Given a feasible solution, the presolve method uses a heuristic based on selectively fixing variables based on column pricing information. Initially, all columns are priced out and sorted based on the pricing. Those within a selected percentage of the best potential columns remain unfixed (for example, the top 10 percent), whereas the remaining ones are fixed. We presolve to reduce the problem size substantially and then solve the LP using the primal simplex method. Once we determine the optimal solution for the reduced subproblem, we unfix variables, recompute the column prices, select a new set of variables, and fix the remaining unselected variables. As before, this new model is presolved and then solved via the simplex method. The procedure continues iteratively until (a) the relative improvement in the objective function from one iteration to the next falls below a certain threshold or (b) a maximum number of iterations is reached. Subsequent to meeting the stopping criteria for the presolve heuristic, we solve the complete model to optimality with the advanced basis.

For the presolve heuristic, we experimented to determine the appropriate number of target columns to choose in each iteration as well as the maximum number of iterations. Once calibrated, the heuristic works well (and is quite robust) for SCO models because of the model structure. Large portions of these types of SCO models are made up of long sections of straight BOM, with a limited number of alternate manufacturing processes for building a given assembly PN. As a result, explicitly fixing some decision variables based on column prices and subsequently presolving the model result in many decision variables over several time periods being removed, greatly reducing the model size at each iteration.

The presolve heuristic reduces computation time. Instead of solving one instance of the model each day, users can now solve several instances per day. They use our model for what-if analysis and try-for-fit applications as well as for daily planning. They analyze the solution to one model (for example, for capacity bottlenecks or supply shortages) and develop recourse options (for example, outsourcing or short-term leasing of tools, rescheduling of supply deliveries) that they can incorporate in the next instance of the model.

Demand Prioritization

An early implementation of an LP model for semiconductor SCO at IBM treated all demand as equally important. Planners prioritized demand using manual adjustments to supply chain plans. Initially users rejected the idea of considering demand classes within the MIP. However, users were eventually convinced that explicit modeling of a small number of demand classes could (potentially) be an effective approach for trading off supply and demand matching decisions. Initially user prototyping began with just two demand classes. However, once users saw the value of modeling demand classes explicitly, they encouraged us to increase the number of demand classes.

In the current implementation of the MIP, we assign demands different priorities by assigning a demand-class subscript to variables F_{tmakq} and B_{tmkq}. For example, demand class 1 may be associated with customer orders that have been committed and are scheduled to be filled in the near term (for example, in four weeks). Other demand classes may be associated with order reservations, buffers, or safety stocks and hence have lower priorities because they do not support an actual customer order. We indicate the varying priorities by applying back-order costs, BC_{tmkq}, that are decreasing in demand class q.

Ideally, the method we used for modeling back-order costs would guarantee demand-class compliance for any number of demand priorities for a single model instance; that is, it would allocate capacity and inventory assets to satisfy higher-priority demands (to the extent possible) before lower-priority demands. In theory, we could do this by setting the back-order penalty for a particular demand-class priority arbitrarily higher than the penalty for the next lower-demand priority. However, in practice, because the numerical accuracy of floating point operations on computers is finite, there are bounds on the range of objective function penalties for which LP factorization

methods are stable (experimental evidence implies that 0.01 to 1,000,000 is a reasonable assumption when operating with double precision using 64-bit compilation). Therefore, the bounds on the differences between objective function coefficients represent adjacent demand classes. These bounds effectively limit the number of demand classes that we can model accurately within the MIP.

The rapid proliferation of demand classes by users of the MIP at IBM led to eventual deterioration in demand-class compliance within the MIP model. In response to this problem, we have developed a method for dealing with this practical issue based on consideration of groups of demand classes. The method iteratively modifies and solves the respective MIPs for each group and leverages the results from one solution to warm start the next. The combination of grouping and warm starting saves computation time while respecting demand classes in the solution. The method considers relaxed versions of the MIP model at each iteration that allow for flexibility in realigning resources (for example, work-center capacity) to accommodate low-priority groups without sacrificing higher-priority groups. Within each group, back-order costs are calibrated, based on a function that describes the relative importance of demand classes, subject to constraints on the minimum and maximum allowable cost coefficients in the MIP.

Preemptive Priority Algorithm

Step 1. Define demand-class groups from the set of demand classes and index the groups from $i = 1, \ldots, n$ from the lowest to the highest priority. The number of distinct demand classes in group i is $N(i)$. Set $i = 1$.

Step 2. Set $F_{tmakq} = 0 \ \forall q > \sum_{k=1}^{i} N(k)$ and unfix all F_{mtakq}, such that q is in the current priority group or a lower (more important) priority group than i.

Step 3. Recalibrate cost penalties for variables B_{mtkq} for all q in group i, making use of the full allowable range of cost coefficients.

Step 4. Solve the modified group i LP: If $(i > 1)$; then warm start the LP solution using the previous solution as an advanced basis.

Step 5. Add the following new constraint requiring that B_{mtkq} variables in group i are lower bounded, based on the back-order variables, B^*_{mtkq}, in the current LP solution from Step 4:

$$\sum_{m,k} B_{mtkq} \leq \sum_{m,k} B^*_{mtkq} \quad \forall t, q.$$

Step 6. Reset the LP basis with the current variables fixed using the dual simplex method. Increment group to $i = i + 1$. If $(i = n + 1)$ stop; otherwise return to Step 2.

Discrete Production Releases

Lot sizing is important because silicon wafers are manufactured in discrete containers. In an early implementation of an LP model for supply chain planning at IBM, lot sizing was not considered. However, because lot sizes were reflected in actual execution of production (through manual adjustments to plans), this meant that the resulting MIP solution contained inherent accuracies in tool-capacity requirements and in matching finished products with customer demand. The relatively coarse grain of lot sizes with respect to daily tool capacity meant that the model had serious shortcomings in accuracy. The consideration of lot sizing within the MIP model was critical in achieving optimal allocation of supply and capacity throughout IBM's semiconductor supply chain.

We developed a heuristic for achieving feasible and near-optimal solutions to the MIP, given consideration of the discrete constraints $P_{tmae} \in \Xi$, where Ξ represents the discrete set of allowable production starts based on lot-sizing rules. Our heuristic starts with a solution to the root node problem, that is, the LP with the discrete lot-sizing constraints relaxed. We then use a two-step heuristic to determine a feasible and near-optimal solution to the MIP problem. In Step 1, we sequentially modify the production start variables, P_{tmae}, through a depth-first search according to a variable sequencing and branching strategy designed to produce a good feasible solution quickly. In Step 2, we look for local improvements by sequentially relaxing selected lot-sizing constraints in the Step 1 MIP to improve the initial solution.

The concept of a low-level code is well known in the context of MRP (Orlicky 1975). Low-level codes denote levels of the BOM. For example, finished goods with associated customer demand might be assigned low-level code 0; assemblies used to build

these finished products would then be assigned low-level code 1; and so on. While moving through the binary tree, it may be infeasible to branch up because doing so may, for example, violate a production constraint (for example, insufficient work-center capacity to increase the production start). In such a case, it is necessary to backtrack, that is, reverse the lot-size decision and branch down. Furthermore, the potential for backtracking is not limited to a single iteration. For example, backtracking one iteration would not be sufficient because rounding down could also be infeasible. We might use multiple-iteration backtracking when a previously lot-sized variable for an assembly has been fixed, and therefore it is infeasible to branch down a variable representing a component to the assembly. In practice, when a BOM has many levels, this backtracking can mean unacceptable computation times. However, sorting the variables from highest to lowest low-level code in the BOM (for example, raw materials to finished products) limits the backtracking to a single iteration at most. Sorting guarantees that no assembly variable is rounded before its associated components, that is, we can always round the assembly production start variable down.

We based the branching strategy in Step 1 on implicitly representing the branch-and-bound tree as a set of discrete feasible values defined by the nearest neighbor subset of Ξ, rather than the alternative of explicitly reformulating the LP to include binary variables and additional constraints. In addition to allowing for a much more manageable LP relaxation at each node, this strategy also makes it straightforward to selectively relax lot-sizing rules for very large production starts when it is a reasonable approximation to ignore lot sizing. The discrete feasible values are represented by a binary tree. The binary tree consists of nodes representing relaxed LPs to be solved, each branching to two other nodes, denoting a variable's branching up or down for a given variable to the next higher or lower feasible lot-sized values. In reality, this use of a binary tree is an approximation because a given variable may take on a range of possible values (for example, for a container constraint any integer multiple of container size is feasible). To avoid excessive restriction of the set of feasible values, we recompute the nearest neighbor subset of Ξ dynamically as progress is made through the binary tree in Step 1 and consider only nearest neighbors as candidates to be successor nodes. Therefore, the binary tree definition is dynamically changing as the heuristic proceeds. This method effectively trades off the needs for high-quality solutions and short computation times.

In Step 1, we preferentially branch variables up to the next feasible lot-sized value in Ξ, provided the resulting LP has a feasible solution. This branching strategy is consistent with a high-service-level commitment that is typical in the semiconductor industry. An exception to the decision to branch up is applied if the change in the objective function between the original LP (prior to branching) and the new LP (after branching) exceeds a user-defined tolerance. This could happen, for example, if increasing the current production-start variable would steal capacity from a production start (for a more important demand class). The Step 1 heuristic also allows us to branch multiple (n) variables at a time, which can improve computational efficiency. For each set of n variables in Step 1, we can decide to branch up or down. Either option is permissible, but both may not be feasible because of previous branching decisions. In the event that branching on n variables is infeasible, the heuristic reverts to branching one variable at a time.

In Step 2, a series of iterations is conducted to improve the quality of the solution obtained in Step 1. The solution is successively improved by partitioning the problem into subproblems, based on related parts in the BOM, and computing improved local solutions for each subproblem. The search iterates through subproblems until either the solution improvement for a pass through the subsets is below some user-defined tolerance or the total run time exceeds the allotted time specified by the user (for example, one hour). Each subset of variables has its previous branching decisions relaxed, and a search is carried out to determine whether a different lot sizing of the variables in the subset would improve the solution. In general, any search method may be used to solve the restricted MIPs in Step 2. We find it effective to use a modified version of our Step 1 method, sequencing variables in Substep (iii) in increasing order of low-level code (rather than decreasing order). We thus diversify the portions of the branch-and-bound tree that are explored, resulting in the greater potential for solution improvements.

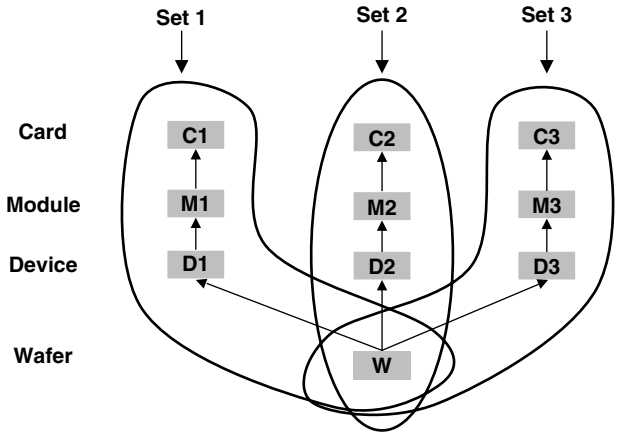

Figure 2: This diagram illustrates how the lot-sizing heuristic groups part numbers in the bill of material to decompose the mixed-integer program into smaller subproblems. The solution of smaller subproblems is the basis for the fast improvement heuristic for discrete lot sizing of semiconductor production starts.

In general, any approach for allocating variables to sets can be used in Step 2. The specific method that we have used (Figure 2) involves partitioning P_{tmae} variables based on their associated PNs using the BOM. Each iteration begins with a finished-good part at the top of the BOM, that is, parts with external demand that are not components of any other part. A breadth-first search is carried out, and all parts connected through the BOM to the finished-good part are grouped into the same set. Once all connected parts are collected, the iteration is complete and a new iteration begins with the next unprocessed finished-good part. The process continues until all parts have been allocated to a unique subset.

The Two-Step Lot-Sizing Heuristic

Step 1. In Step 1, the initial lot-sized solution construction, perform a depth-first search to iteratively adjust variable values so that they satisfy discrete lot-size constraints (Figure 3):

(i) Solve the root node LP using a primal-simplex-based method and maintain this solution as memory resident.

(ii) Fix all variables with associated lot-size constraints that have values within some ϵ tolerance of feasible lot-sized values.

(iii) Obtain the next lower and higher feasible lot-sized quantities in Ξ and sort the pending variables, P_{tmae}, in a list by increasing order of period, j, decreasing order of low-level code position in the BOM for the part, m, and increasing in difference between the

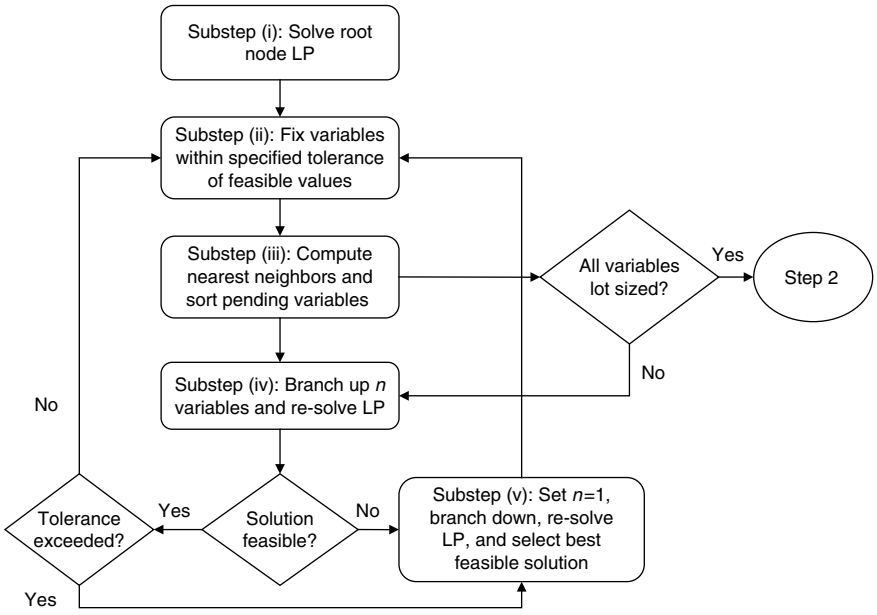

Figure 3: We show the steps involved in the initial construction of a feasible solution to the mixed-integer program constituting the first step of our heuristic for acquiring a solution with lot-sized production starts.

current value and the nearest higher feasible lot-sized value. If all variables satisfy lot-sizing constraints, then proceed to Step 2.

(iv) Choose the first n variables in the sorted list, branch up the n variables, and re-solve the resulting LP using the dual-simplex method. If the LP has a feasible solution and the change in the objective function does not exceed a user-defined tolerance, then return to Substep (ii).

(v) If $n > 1$, set $n = 1$, relax branching constraints for the chosen n variables, and return to Substep (ii). Otherwise, branch down on the variable and re-solve the resulting LP.

(vi) Select the best solution from the up and down branches and return to Substep (ii).

Step 2. In Step 2, the lot-sized solution improvement, iteratively compute revised solutions to separable subproblems, as follows (Figure 4):

(i) Construct subsets of P_{tmae} variables, $\mathcal{S}_i$, $i = 1, \ldots, L$, based on the BOM structure (Figure 2). Set $i = 1$.

(ii) If $i = L$, set $i = 1$. Otherwise, set $i = i + 1$. For variables in subset $\mathcal{S}_i$, relax branching constraints added in Step 1 or Step 2.

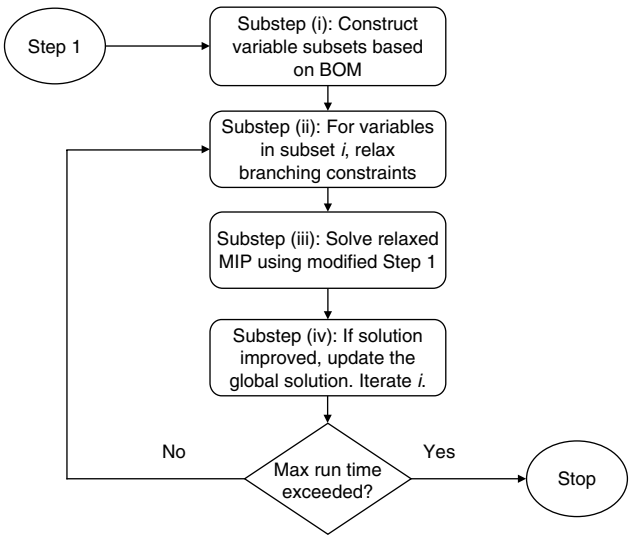

Figure 4: The steps involved in the improvement of the initial feasible solution to the mixed-integer program make up the second step of our heuristic, in which the lot-sized production starts are iteratively improved to generate a final and near-optimal solution.

(iii) Solve the resulting MIP using the method of Step 1 but with variables reverse-sequenced in order of increasing low-level code.

(iv) If an improved solution results from Substep (iii), then update the global solution with the local improvement.

(v) If the specified maximum run time has been exceeded, then stop with the current solution (stopping condition). Otherwise, return to Substep (ii).

IBM has deployed this lot-sizing heuristic successfully in a production application of the MIP model. The most significant hurdles in implementing the method were due to the heuristic nature of the solution. Whereas solutions to the LP relaxation of the MIP model are optimal, solutions based on the lot-sizing heuristic are not necessarily optimal. In practice, the heuristic provides solutions that are, in aggregate, close to optimal. However, it is possible to find specific cases that are suboptimal. Our careful explanation of the complex nature of the problem (and the limitations on all heuristics) and our demonstration that our solutions are typically near optimal contributed to user acceptance and to the successful implementation of our method.

Impact

The MIP is a critical part of the CPE, which has been used extensively in the IBM Systems and Technology Group in planning semiconductor supply chain operations. Planners solve instances of the CPE many times each day to create enterprisewide supply chain plans. In particular, they use the CPE to

—Calculate a detailed worldwide supply that is available for customer commitments,

—Provide production and shipping directions to all manufacturing lines in the enterprise,

—Optimize material and capacity allocations,

—Execute what-if and other try-for-fit sizings, and

—Facilitate strategic planning and operational planning.

The CPE and associated changes to IBM's business processes have

—Improved on-time deliveries by 15 percent,

—Improved asset utilization by two to four percent of costs, and

—Reduced inventory by 25 to 30 percent.

The CPE has improved customer service, inventory requirements, and asset utilization. Instead of determining an optimal point on a service-inventory tradeoff curve, the CPE has enabled a shift in the entire curve. IBM uses the MIP model for planning about a third of the semiconductor parts processed by the CPE and handles the other parts with fast heuristic algorithms. The parts it plans using the MIP tend to be the most complex from a supply chain perspective, for instance, they have more complicated material flows and capacity utilization (for example, binning, multiple production processes, and processing at multiple sites). In the semiconductor industry, the parts that are most complicated to plan are typically those producing the highest revenue. Therefore, whereas the MIP is associated with approximately one-third of all semiconductor parts planned at IBM, the total associated revenue is at least one-third of the total. Therefore, the MIP is associated with a significant portion of the financial and customer-service impact described.

In addition to the quantitative benefits, there are important impacts of our heuristics that are difficult to measure directly. For instance, the presolve heuristic has resulted in substantial reduction of computation time, to the point where a CPE model of the entire enterprise can be solved in less than three hours; it has reduced MIP computation time by 75 percent. In practice, models such as the CPE are limited by the number of scenarios that can be successfully analyzed within a fixed cycle time for the overall business process. Because of reductions in computation time, IBM planners can analyze several scenarios per day, responding to capacity and supply bottlenecks in the supply chain, and feeding back recourse options into the next instance of the model. Furthermore, preemptive demand-class prioritization allows IBM to plan for true demand priorities. With these OR innovations, IBM can allocate the assets of its multibillion dollar enterprise intelligently and rapidly and simultaneously improve its profitability and customer service.

Appendix

Following is a detailed description of the MIP model:

Indices

t: Time period index $1, \ldots, T$.
m: Material (part number) from $1, \ldots, M$.
a: Plant location within the enterprise $1, \ldots, A$.
n: Material being substituted $1, \ldots, N$.
e: Process used to make the material $1, \ldots, E$.
v: Receiving plant location $1, \ldots, V$.
u: Consuming location (may be plant or demand center $1, \ldots, U$).
k: Customer demand center $1, \ldots, K$.
q: Demand class (relative priority) $1, \ldots, Q$.
w: Resource capacity $1, \ldots, W$.
z: Collection of related parts based on sourcing rules $z = 1, \ldots, Z$.
CT: Set relating cycle time in period x to the current period t, where $CT = \{x \mid x = t - ct_{xmae}\}$.
TT: Set relating transit time in period x to the current period t, where $TT = \{x \mid x = t - tt_{xmav}\}$.
AC_{mn}: Set of parts m that are components of assembly n.

Decision Variables

I_{tma}: Inventory at the end of period t for part m at plant a.
P_{tmae}: Production starts of part m during period t at plant a using process e.
L_{tmna}: Amount of part n substituted by part m during period t at plant a.
T_{tmav}: Internal shipments of part m leaving plant a during period t destined for plant v.
F_{tmakq}: Shipments of part m leaving plant a during period t satisfying class q demand at customer k.
B_{tmkq}: Back orders of part m at the end of period t for class q demand at customer location k.
H_{jzu}: Total shipments of group z parts leaving suppliers during period t to location(s) u.
S_{jzau}: Shipment of part group z from plant a to u during period t that exceeds the maximum target.
G_{jzau}: Shipment of part group z from plant a to u during period t that falls short of the minimum target.

Model Constants

PC_{tmae}: Cost of releasing one piece of part m during period t at plant a using process e.
LC_{tmna}: Substitution cost per piece of part n substituted by part m in period t at plant a.
TC_{tmav}: Transportation cost per piece of part m leaving plant a during period t destined for plant v.

SC_{tmak}: Shipping cost per piece of part m leaving plant a during period t destined for customer k.
IC_{tma}: Inventory holding cost per piece of part m in period t at a plant a.
BC_{tmkq}: Back-order cost per piece of part m in period t for class q demand at customer k.
CS_{tzau}: Cost of deviating from sourcing rule in period t for consuming location u and part group z.
D_{tmkq}: Demand requested in period t for part m at customer location k for demand class q.
R_{tma}: Receipts (projected WIP or purchase orders) of part m to be received at plant a in period t.
U_{taw}: Capacity of resource w available at plant a during period t.
V_{tmaew}: Capacity of resource w required per piece of part m at plant a for process e in period t.
Z_{tmaen}: Amount of component m per assembly n started in period t at plant a with process e.
Y_{tmae}: Output of part m per piece started at plant a in period t using process e.
E_{tmna}: Amount of part m required to substitute per piece for part n at plant a in period t.
ct_{tmae}: Cycle time for production of part m at plant a using process e in period t.
tt_{mav}: Transit time for part m from plant a to plant v.
Ξ: Discrete set of feasible production starts based on lot-size constraints.

MIP Model

$$\min\Bigg\{\sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{a=1}^{A}\sum_{e=1}^{E} PC_{tmae}P_{tmae} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{a=1}^{A} LC_{tmna}L_{tmna} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{a=1}^{A}\sum_{v=1}^{V} TC_{tmav}T_{tmav} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{a=1}^{A} IC_{tma}I_{tma} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{q=1}^{Q} BC_{tmkq}B_{tmkq} + \sum_{t=1}^{T}\sum_{m=1}^{M}\sum_{a=1}^{A}\sum_{k=1}^{K}\sum_{q=1}^{Q} SC_{tmak}F_{tmakq} + \sum_{t=1}^{T}\sum_{m\in Z}\sum_{a=1}^{A}\sum_{u=1}^{U} CS_{tzau}(S_{tzau}+G_{tzau})\Bigg\},$$

s.t.

$$\text{(i)}\quad B_{tmkq} - B_{t-1mkq} + \sum_{a=1}^{A} F_{tmakq} = D_{tmkq} \quad \forall\, t, m, k, q,$$

$$\text{(ii)}\quad I_{tma} - I_{t-1ma} - \sum_{t\in CT} Y_{tmae}P_{xmae} + \sum_{n} L_{tmna} - \sum_{n} E_{tmna}L_{tnma} - \sum_{t\in TT}\sum_{v=1}^{V} T_{tmva} + \sum_{v=1}^{V} T_{tmav} + \sum_{k=1}^{K}\sum_{q=1}^{Q} F_{tmakq} + \sum_{n\in AC} Z_{tmaen}P_{tnae} = R_{tma} \quad \forall\, t, m, a,$$

$$\text{(iii)}\quad \sum_{m=1}^{M}\sum_{e=1}^{E} V_{tmaew}P_{tmae} \le U_{taw} \quad \forall\, t, a, w,$$

$$\text{(iv—a)}\quad H_{tzu} - \sum_{m\in z}\sum_{a=1}^{A}\left(T_{tmau} + \sum_{q=1}^{Q} F_{tmauq}\right) = 0 \quad \forall\, z, t, u$$

$$\text{(iv—b)}\quad \sum_{m\in z}\left(T_{tmau} + \sum_{q=1}^{Q} F_{tmauq}\right) - S_{tzau} - \max_{tzau} H_{tzu} \le 0 \quad \forall\, z, t, a$$

$$\text{(iv—c)}\quad \sum_{m\in z}\left(T_{tmau} + \sum_{q=1}^{Q} F_{tmauq}\right) + G_{tzau} - \min_{tzau} H_{tzu} \ge 0 \quad \forall\, z, t, a,$$

$$\text{(v)}\quad P_{tmae} \in \Xi,$$

$$P_{tmae}, L_{tmna}, F_{tmakq}, T_{tmav}, I_{tma}, B_{tmkq} \ge 0.$$

References

Arntzen, B. C., G. G. Brown, T. P. Harrison, L. L. Trafton. 1995. Global supply chain management at Digital Equipment Corporation. *Interfaces* **25**(1) 69–93.

Belvaux, G., L. A. Wolsey. 2000. *bc-prod*: A specialized branch-and-cut system for lot-sizing problems. *Management Sci.* **46**(5) 724–738.

Belvaux, G., L. A. Wolsey. 2001. Modelling practical lot-sizing problems as mixed-integer programs. *Management Sci.* **47**(7) 993–1007.

Bermon, S., S. J. Hood. 1999. Capacity optimization planning system (CAPS). *Interfaces* **29**(5) 31–50.

Camm, J. D., T. E. Chorman, F. A. Dill, J. R. Evans, D. J. Sweeney, G. W. Wegryn. 1997. Blending OR/MS, judgment, and GIS: Restructuring P&G's supply chain. *Interfaces* **27**(1) 128–142.

Dantzig, G. B., P. Wolfe. 1960. Decomposition principle for linear programs. *Oper. Res.* **8**(1) 101–111.

Hegde, S. R., R. J. Milne, R. A. Orzell, M. C. Patil, S. P. Patil. 2004. Decomposition system and method for solving a large scale semiconductor production planning problem. US Patent 6,701,201.

Kuik, R., M. Salomon, L. N. Van Wassenhove. 1994. Batching decisions: Structure and models. *Eur. J. Oper. Res.* **75**(2) 243–263.

Leachman, R. C., R. F. Benson, C. Liu, D. J. Raar. 1996. IMPREeSS: An automated production-planning and delivery-quotation system at Harris Corporation–Semiconductor Sector. *Interfaces* **26**(1) 6–37.

Lin, G., M. Ettl, S. Buckley, S. Bagchi, D. D. Yao, B. L. Naccarato, R. Allan, K. Kim, L. Koenig. 2000. Extended-enterprise supply-chain management at IBM personal systems group and other divisions. *Interfaces* **30**(1) 7–25.

Lyon, P., R. J. Milne, R. Orzell, R. Rice. 2001. Matching assets with demand in supply-chain management at IBM Microelectronics. *Interfaces* **31**(1) 108–124.

Orlicky, J. 1975. *Materials Requirements Planning: The New Way of Life in Production and Inventory Management*. McGraw-Hill, New York.

Tayur, S., R. Ganeshan, M. Magazine, eds. 1998. *Quantitative Models for Supply Chain Management*. Kluwer Academic Publishers, Boston, MA.

Uzsoy, R., C. Lee, L. A. Martin-Vega. 1992. A review of production planning and scheduling models in the semiconductor industry, Part I: System characteristics, performance evaluation and production planning. *IIE Trans. Scheduling Logist.* **24**(4) 47–60.

Uzsoy, R., C. Lee, L. A. Martin-Vega. 1994. A review of production planning and scheduling models in the semiconductor industry, Part II: Shop floor control. *IIE Trans. Scheduling Logist.* **26**(5) 44–55.

Wagner, H., T. Whitin. 1958. Dynamic version of the economic lot size model. *Management Sci.* **5**(1) 89–96.

Wolsey, L. A. 2002. Solving multi-item lot-sizing problems with an MIP solver using classification and reformulation. *Oper. Res.* **48**(12) 1587–1602.