Finite Production Planning Using a Spreadsheet Sifter

Edmund W. Schuster
Welch's
Concord, MA

Stuart J. Allen
Penn State Erie, The Behrend College
Erie, PA

The study of finite planning systems takes place from several different perspectives at the *Center for Process Manufacturing* located on the campus of Penn State Erie, The Behrend College. First, we formulate practical mathematical models to solve specific finite planning problems within the process industries. Second, we seek to understand and categorize the vast amount of published research on finite planning systems. Finally, we desire to communicate to practitioners the different techniques of finite planning with data drawn from industry.

This article describes a simple approach to finite planning involving a combination of simulation and optimization. We will demonstrate the method using actual demand and capacity data from a production line at Welch's. A solution is obtained using spreadsheet software.

The Finite Planning Problem

Process oriented firms usually have high speed manufacturing lines that produce a fixed number of end items. Production planners play an important role in plant operations by scheduling the sequence of end item production to meet the demand forecast, while taking into account many factors such as customer service levels, forecast bias, manufacturing lead time, capacity, inventory carrying cost, set-up cost and lot sizing. Complicating matters, the consumer goods segment of the process industries often deals with dynamic demand caused by frequent use of trade promotion. For many consumer goods manufacturers, it is common to sell 90% of yearly demand for key items during tightly focused drive periods. The resulting lumpy demand pattern proves a difficult problem in production planning.

Spreadsheets now offer enough simulation and mathematical programming capability to build models that accomplish finite production planning in a lumpy demand environment. This opens a wide range of new possibilities for solving finite production planning problems. Despite the power of microcomputer technology, spreadsheets remain underutilized as an inexpensive method of finite planning in the process industries.

The model presented in this article combines a deterministic simulation previously discussed by Schuster & Finch [1990], and an integer programming model formulated by Dzielinski & Gomory [1965], with later expansion by Nahmias [1989, p. 115-118]. During the course of our research, we observed interesting synergies occurred by blending the deterministic simulation model together with the integer programming model. This "blending of models" leads to new ways of looking at the finite planning problem. The model we now present has the simple purpose of planning production of end items produced on a manufacturing line operating in a dynamic demand environment.

As with any model, the first step in its understanding begins with a discussion of underlying assumptions. The spreadsheet model described in this article assumes the following:

1. A fixed number of items are run on a dedicated production line under a make to stock strategy. A forecast of each end item exists and is used for production planning.

2. In the examples to follow, finite production planning occurs in weekly time buckets with an eight week horizon. The output of the model is a least cost weekly production plan that meets finite capacity limits.

3. Switching from one end item to another requires a major changeover. The time for each changeover is fixed during the planning horizon.

4. Inventory carrying costs and changeover costs are known.

5. Manufacturing line capacity and changeover capacity limits are known.

With these assumptions in mind, we now turn our attention to discussion of the spreadsheet model.

Simulating Production Vectors

The deterministic simulation uses a time phased re-order point method to calculate the production spacing required to satisfy buffer stock requirements. Dynamic in nature, the buffer stocks depend on the demand forecast as well as other important factors such as customer service, production lead-time and forecast bias.

The deterministic simulation served as a useful production planning tool at Welch's for many years. However, it had major shortcomings in dealing with capacity constraints. Production plans developed for each end item were capacity infinite, and independent of other items produced on the manufacturing line. This caused frequent capacity violations and ineffective production plans. In addition, production plans from the deterministic simulation failed to consider set-up or holding cost, and provided no total cost optimization.

The cure to this problem begins by calculating a set of production plans for each item using the deterministic simulation. Adjustment of several simulation parameters results in plans with different production spacing.

For example, one way of obtaining different production plans for a single end item involves varying the fixed lot size used for each production run over the planning horizon. Large lot sizes mean production takes place less frequently and average inventory remains high. On the other hand, small lot sizes mean frequent production and low inventories. By varying the lot size in natural increments such as half shifts of production, a set of production plans results. Dzielinski & Gomory refer to each alternative production plan as a *vector*.

Each *vector* has a different cost associated with it. Costs are calculated by analyzing the number of changeovers and the average inventory level associated with a particular *vector*. Table 1 provides an example of a set of *vectors* for an item.

Please place Table 1 about here

In Table 1 we observe four *vectors* associated with product code 116. This product code represents a frozen grape concentrate item produced by Welch's and sold in retail stores. The first *vector* (labeled as *vector* 1) represents a one-half shift production strategy. With small lot sizes, production occurs frequently to meet the demand forecast.

The cost of the one-half shift lot size depends on the number of changeovers required for production, and the average inventory level resulting from the spacing of production. If we assume set-up cost equals $500 and inventory carrying cost equals $0.14/unit/week, then total cost becomes:

[# of set-ups] x [cost per set-up] = set-up cost

7 set-ups x $500/set-up = $3,500

and,

[average inventory] x [inventory carrying cost per week] = inventory cost

12,800 units x $0.14/unit/ week = $1,791

[set-up cost] + [inventory cost] = total cost

$3,500 + $1,792 = $5,291

From Table 1, we notice that the two shift lot size, *vector 4,* has the least cost of all *vectors*. The large lot size causes high average inventories, but few expensive changeovers. If product code 116 was the only item run on the manufacturing line, we would pick the two shift lot size as the least cost alternative.

A more complex situation arises with several items produced on a manufacturing line. The two shift lot size for product code 116 may cause capacity conflicts with the least cost *vector* for other items produced on the manufacturing line. Furthermore, we have only considered actual manufacturing time in our calculation of capacity. Set-up time is also subject to capacity limits.

Somehow we must sort through all *vectors* to find the best mix that minimizes cost while satisfying production and set-up capacity limits. Integer programming using binary variables provides a simple method to sort through different *vectors*. Our discussion will now focus on a spreadsheet based model that selects the least cost set of *vectors*.

The Sifter

An integer program can act like a sifter by choosing the single least cost *vector* for each end item that collectively meets production and set-up capacity limits. The sifting action occurs from using binary decision variables for each *vector*. An example provides the best way to understand the sifting action.

Suppose we arrange the set of four *vectors* associated with product code 116 vertically on a spreadsheet. Next to the set of *vectors* for product code 116, we add five sets of four *vectors* representing other products run on the manufacturing line (see Table 2, product code 116 shown in bold).

Please place Table 2 about here

In Table 2, we manually entered a row of 1's and 0's designating selection of a *vector*. When a 1 appears in this row, the corresponding *vector* becomes part of the weekly production plan. If a zero appears, the production plan does not include the *vector*. The weekly production plan includes one *vector* per item. Spreadsheet formulas called *vector* products multiply the row of 1's and 0's by the production or set-up capacity consumed for each *vector*, in each time period, to arrive at total *capacity requirements* per week.

Summing the cost of all chosen *vectors* gives the *total cost* of the production plan. *Capacity utilization* follows through division of *capacity requirements* by the *capacity limit*. To account for set-up time, we extend the *vector* to show set-up hours associated with each production run. This allows separate limits on production capacity and set-up capacity.

As an initial try at a feasible solution, we manually selected the least cost *vector* for each item in Table 2 and computed total production time, set-up time and cost. Table 3 shows the group of least cost *vectors* selected from Table 2. The cost of the production plan equals $22,693. However, we exceed production capacity in weeks 1 and 5.

To get a feasible solution, we may try manually selecting another combination of *vectors*. Deciding which *vectors* to choose becomes a problem. In this example, there are 4096 possible combinations of *vectors* to make up a production plan. Only by trying all combinations of *vectors* can we know the least cost mix of *vectors* that meet capacity limitations.

Please place Table 3 about here

Using Spreadsheet Optimization to Sift Vectors

Rather than attempting all the combinations of *vectors*, we can use integer programming to mathematically sift through all possible *vector* combinations and arrive at the best solution. Several software packages offer integer programming capability in a spreadsheet environment. We chose **What's Best!** (distributed by LINDO SYSTEMS) which works as an add-on to Microsoft Excel and Lotus 1-2-3. Commands for **What's Best!** work from easy to use, pull down menus. Because **What's Best!** overlays a spreadsheet, managers find it easy to apply mathematical programming to production and inventory management problems. For a complete description on how to use **What's Best!** please refer to a recent book authored by Plane [1994].

For smaller problems, Microsoft Excel has a "solver" contained as part of the spreadsheet. The solver can do integer programming and is listed under the "tools" menu (please note that in order to activate "solver" in your spreadsheet, you may need to specify "solver" under the add-ins option of the tools menu). In the next PI - SIG news letter, we will show the strengths and weaknesses of using "solver" to find a solution to the sifter problem.

The spreadsheet appearance of the integer programming problem closely resembles the layout of Table 2. The row of 1's and 0's serve as decision variables. When What's Best solves the integer programming problem, 1's and 0's indicate which *vectors* make up the optimal solution (1=accept, 0=reject).

Each row in Table 2 serves as a constraint. The right hand side of each row must be less than the weekly *capacity limit*. Under circumstances of high capacity utilization, it may be possible that no combination of *vectors* meets the capacity limit for production time or set-up time.

To guard against this dilemma, several modifications of the constraints allow for overtime at a cost penalty. With the objective to minimize cost, the integer program seeks all possible combinations of *vectors* not causing overtime. If no combination of *vectors* meets capacity limitations, the model chooses the closest fit of *vectors* that results in planned overtime. For a complete description of the mathematical formulation for the sifter, please refer to the appendix.

Using **What's Best!**, we solved the finite capacity problem from Table 2. The solution appears in Table 4. Notice capacity violations no longer exist in weeks 1 and 5. However, the new solution has a slightly higher total cost ($22,776 as compared to $22,693 for our initial solution).

Please place Table 4 about here

As an exercise, we ask readers to constrain the scheduling problem outlined in table 2 by reducing production time to 50% of normal capacity during weeks 5 and 6. See if you can solve this scheduling problem without the use of integer programming. We will publish the answer to this question in the next issue of the PI SIG newsletter.

Conclusion

With spreadsheet simulation and optimization tools, practitioners can build effective finite production planning models that help in understanding the power of mathematics to solve practical problems encountered by industry. At the *Center for Process Manufacturing*, we strive to bring the ideas of mathematics to practice, and to promote the general use of models in business problem - solving. We hope that by our research the members of the PI - SIG of APICS will gain greater insights into the underpinnings of finite planning systems.

References

Dzielinski, B.C., and R.E. Gomory, 1965, "Optimal Programming of Lot Sizes, Inventory and Labor Allocations," Management Science 11, pages 875-90.

Nahmias, S., 1989, Production and Operations Analysis, Richard D. Irwin, Inc., Boston, MA.

Plane, D. R., 1994, Management Science: A Spreadsheet Approach, Boyd & Fraser, Danvers, MA.

Schuster, E.W., and B.J. Finch, 1990, "A Deterministic Spreadsheet Simulation Model for Production Scheduling in a Lumpy Demand Environment," Production and Inventory Management Journal, Volume 33, Number 1.

Appendix

1. A note concerning vectors.

The *vectors* presented in this article result from a deterministic spreadsheet simulation discussed by Schuster and Finch. However, this is not the only method available for calculating a set of *vectors*. To simplify the model, practitioners can use a time phased reorder point with fixed safety stock. Another method to quickly generate *vectors* involves using the exact requirements policy [Nahmias 1989, p. 115].

2. Mathematical formulation of the sifter.

The sifter is an integer program with the objective function restricted to binary variables.

i = product code

j = *vector* of production quantities for product i

t = weekly time periods

C(i,j) = Cost of producing item i using production *vector* j.

CR(t) = Production run time capacity limit for time period t

CS(t) = Set-up time capacity limit for time period t

r(i,j,t) = production time required for product code i using production *vector* j in period t

s(i,j,t) = set-up time required for product code I using production *vector* j in time period t.

H = Hours per shift (in our examples, we assume 8 hrs. per shift for production and set-up)

MP = production overtime cost

MS = set-up overtime cost

M = overtime capacity (production time + set-up time) for time period t

Objective Equation:

$$Min \sum_{i=1}^{I} \sum_{j=1}^{J} C(i,j)\theta(i,j) + \sum_{t=1}^{T} [MPe(t) + MSd(t)]$$

Subject to:

1. Production time Constraint

$$\sum_{i=1}^{I} \sum_{j=1}^{J} r(i,j,t)\theta(i,j) - He(t) \leq CR(t) \text{, for all t}$$

2. Set-up time constraint

$$\sum_{i=1}^{I} \sum_{j=1}^{J} s(i,j,t)\theta(i,j) - Hd(t) \leq CS(t) \text{, for all t}$$

3. Overtime Constraint

$$e(t) + d(t) \leq M \text{, for all t}$$

4. Constraint limiting vectors to one per item

$$\sum_{j=1}^{J} \theta(i,j) = 1 \text{, for all i}$$

Where:

$\theta = 1$ or 0, θ is called a sifting variable

e(t) = production overtime for all i

d(t) = set-up overtime for all i

SPECIAL NOTE: Constraint 3 becomes necessary to place realistic limits on set-up and production overtime. In a high capacity utilization situation requiring overtime, constraint 3 may cause a non feasible solution. For this case, generation of additional *vectors* may result in a feasible solution.

Table 1 - Set Of Feasible Production Plans For Product Code 116 (hrs per week)

Vector #	Lot Size	Cost	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Number of Set-ups	Average Inventory (1000's cases)
1	0.5 shift	$5,291	4	4	4	4	0	4	4	4	7	12.8
2	1.0 shift	$4,146	8	0	8	0	0	8	0	8	4	15.3
3	1.5 shift	$4,236	12	0	0	12	0	0	0	12	3	19.6
4	2.0 shift	$4,091	16	0	0	0	0	16	0	0	2	22.1

Table 2 - Complete Set of Vectors

Decision variables	0	0	0	1	0	1	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0	1	0	Total Cost $22,892			
Product Code	105				111				115				122				128				135				Capacity		Capacity	Utilization
Lot Size	0.5 Shift	1.0 Shift	1.5 Shift	2.0 Shift	0.5 Shift	1.0 Shift	1.5 Shift	2.0 Shift	0.5 Shift	1.0 Shift	1.5 Shift	2.0 Shift	0.5 Shift	1.0 Shift	1.5 Shift	2.0 Shift	0.5 Shift	1.0 Shift	1.5 Shift	2.0 Shift	0.5 Shift	1.0 Shift	1.5 Shift	2.0 Shift	Requirement		Limit (hrs)	
Cost	$5,668	$6,339	$6,469	$5,969	$2,075	$2,047	$2,535	$3,522	$5,291	$4,146	$4,236	$4,891	$4,501	$4,324	$3,981	$3,981	$3,122	$2,996	$3,303	$3,279	$4,684	$4,036	$4,011	$4,719				
Production Time (hrs per week)																												
Week 1	0	0	0	0	0	0	0	0	4	8	12	16	8	16	12	16	4	8	12	16	8	8	12	16	48	<	40	120%
Week 2	16	16	24	32	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	32	<	40	80%
Week 3	8	16	0	0	8	16	24	32	4	8	0	0	4	0	0	0	0	0	0	0	0	0	0	0	16	<	40	40%
Week 4	8	0	24	0	0	0	0	0	4	0	12	0	4	0	12	0	4	0	0	0	4	8	0	0	12	<	40	30%
Week 5	8	16	0	32	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	4	0	12	0	44	<	40	110%
Week 6	16	16	24	0	8	0	0	0	4	8	0	16	8	8	0	16	4	8	0	0	4	8	0	16	24	<	40	60%
Week 7	8	0	0	0	0	0	0	0	4	0	0	0	4	8	12	0	4	0	12	0	0	0	0	0	12	<	40	30%
Week 8	16	16	24	32	0	0	0	0	4	8	12	0	8	8	0	16	0	0	0	0	4	0	0	0	32	<	40	80%
Set-up Time (hrs per week)																												
Week 1	0	0	0	0	0	0	0	0	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	32	<	32	100%
Week 2	8	8	8	8	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	<	32	25%
Week 3	8	8	0	0	8	8	8	8	8	8	0	0	8	0	0	0	0	0	0	0	0	0	0	0	8	<	32	25%
Week 4	8	0	8	0	0	0	0	0	8	0	8	0	8	0	8	0	8	0	0	0	8	8	0	0	8	<	32	25%
Week 5	8	8	0	8	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	8	0	8	0	16	<	32	50%
Week 6	8	8	8	0	8	0	0	0	8	8	0	8	8	8	0	8	8	8	0	0	8	8	0	8	16	<	32	50%
Week 7	8	0	0	0	0	0	0	0	8	0	0	0	8	8	8	0	8	0	8	0	0	0	0	0	8	<	32	25%
Week 8	8	8	8	8	0	0	0	0	8	8	8	0	8	8	0	8	0	0	0	0	8	0	0	0	8	<	32	25%

Table 3 - Least Cost Vectors

Total Cost
$22,693

Product Code	103	111	116	122	128	135	Capacity Requirement		Capacity Limit (hrs)	Utilization
Lot Size	2 Shifts	1 Shift	2 Shifts	1.5 Shift	1 Shift	1.5 Shift				
Cost	$5,969	$2,047	$4,091	$3,981	$2,595	$4,011				
Production Time (hrs per week)										
Week 1	0	0	16	12	8	12	48	<	40	**120%**
Week 2	32	0	0	0	0	0	32	<	40	80%
Week 3	0	16	0	0	0	0	16	<	40	40%
Week 4	0	0	0	12	0	0	12	<	40	30%
Week 5	32	0	0	0	0	12	44	<	40	**110%**
Week 6	0	0	16	0	8	0	24	<	40	60%
Week 7	0	0	0	12	0	0	12	<	40	30%
Week 8	32	0	0	0	0	0	32	<	40	80%
Set-up Time (hrs per week)										
Week 1	0	0	8	8	8	8	32	<	32	100%
Week 2	8	0	0	0	0	0	8	<	32	25%
Week 3	0	8	0	0	0	0	8	<	32	25%
Week 4	0	0	0	8	0	0	8	<	32	25%
Week 5	8	0	0	0	0	8	16	<	32	50%
Week 6	0	0	8	0	8	0	16	<	32	50%
Week 7	0	0	0	8	0	0	8	<	32	25%
Week 8	8	0	0	0	0	0	8	<	32	25%

Total Cost
$22,776

Table 4 - Optimal Solution

Product Code	103	111	116	122	128	135	Capacity Requirement		Capacity Limit (hrs)	Utilization
Lot Size	2 Shifts	1 Shift	1 Shift	1.5 Shift	1 Shift	1 Shift				
Cost	$5,969	$2,047	$4,146	$3,981	$2,595	$4,038				
Production Time (hrs per week)										
Week 1	0	0	8	12	8	8	36	<	40	90%
Week 2	32	0	0	0	0	0	32	<	40	80%
Week 3	0	16	8	0	0	0	24	<	40	60%
Week 4	0	0	0	12	0	8	20	<	40	50%
Week 5	32	0	0	0	0	0	32	<	40	80%
Week 6	0	0	8	0	8	8	24	<	40	60%
Week 7	0	0	0	12	0	0	12	<	40	30%
Week 8	32	0	8	0	0	0	40	<	40	100%
Set-up Time (hrs per week)										
Week 1	0	0	8	8	8	8	32	<	32	100%
Week 2	8	0	0	0	0	0	8	<	32	25%
Week 3	0	8	8	0	0	0	16	<	32	50%
Week 4	0	0	0	8	0	8	16	<	32	50%
Week 5	8	0	0	0	0	0	8	<	32	25%
Week 6	0	0	8	0	8	8	24	<	32	75%
Week 7	0	0	0	8	0	0	8	<	32	25%
Week 8	8	0	8	0	0	0	16	<	32	50%