ALLOCATION OF SCARCE RESOURCES IN MANUFACTURING FACILITIES

James H. King

James H. King *joined AT&T in 1985, and is a member of technical staff in the Operations Research Department at AT&T Bell Laboratories in Holmdel, New Jersey. His specialty is inventory management, and his current responsibilities include applying operations research methodology to improve AT&T's manufacturing processes. He has a B.S. in mathematics and economics from Kenyon College in Gambier, Ohio, and a Ph.D. in mathematics from the University of Wisconsin, Madison.*

Factory production schedulers often face a difficult problem after determining that changes in customer orders or critical component availability have affected a production plan's feasibility. They then require a tool to help them determine which of the many feasible schedules is "best." IRAM (the Intraworks Resource Allocation Model) is a personal computer (PC)-based interactive optimization tool that is usable by these planners. It takes advantage of a recently developed resource allocation algorithm that solves large-scale problems very quickly. This paper describes the two main efforts that produced a successful optimization tool from the original theoretical model. First, significant extensions added necessary modeling features to the algorithm while retaining its speed and ability to solve multiperiod planning problems. Second, a user interface was developed to simplify evaluation and change of proposed solutions. As a result, IRAM facilitates the real-time evaluation of numerous alternative schedules.

Introduction

Production schedulers in factories often face a difficult problem when changes in customer orders or critical component availability mean the original planned production schedule is not feasible. The number of these changes is increased by the modern pressure to reduce factory response time to customer orders and the uncertain yields inherent in producing today's high-technology products. If the original schedule cannot be met, planners are responsible for determining how to adjust the schedule to resolve the infeasibilities. However, the large number of alternate feasible schedules makes a manual

Panel 1. Terms and Acronyms in This Paper

BOR	Bill of Resources summarizing the product/resource utilization matrix
FPS	Feasible Production Schedule proposed by IRAM
I/O	Input/Output
IRAM	Intraworks Resource Allocation Model
LP	Linear program
MPS	Master Production Scheduler representing the original production requirements
MRP-II	Material Resource Planning
PC	Personal Computer

adjustment process wasteful. Thus, a tool is needed that can help planners select the schedule that will make optimal use of available resources. Previous attempts to develop production planning tools—usually based on linear programming models—have suffered from long run times and the fact that they did not give the user enough control over the final recommended schedule. Luss and Smith[1] developed an algorithm for these resource allocation problems based on a minimax objective function, which makes the largest weighted deviation between the original and optimal schedules as small as possible. This algorithm offers a potential solution to these drawbacks because it solves large-scale problems very quickly, and because the recommended schedules change smoothly with the model parameters.

This paper describes IRAM (the Intraworks Resource Allocation Model), an interactive production planning tool, based on Luss and Smith's algorithm, that runs on PCs. IRAM helps schedulers find the production schedule that is "best" for them, using their knowledge of the problem. We will describe the two main efforts that were required to produce the tool from the original theoretical model. First, algorithmic extensions were necessary to add features to the final tool without losing the algorithm's original speed. These extensions include a technique for solving multiperiod planning problems and a heuristic method for ensuring integer production values. Second, a user interface was developed that makes it easy for planners to critically evaluate IRAM's proposed solution, and adapt that solution to whatever direction the user feels is necessary.

First, this paper delineates the problem IRAM was designed to solve and identifies its target users. We then discuss IRAM's mathematical algorithm. Finally, we present IRAM's implementation and user interface.

The Problem Statement and the Users

The first requirement for developing a successful optimization tool is understanding the tool's ultimate users and the environment where they work. Ultimately, the success or failure of the tool can depend on how well it responds to its users' needs and constraints.

Factory master-production schedulers are responsible for releasing the production plan (i.e., the master-production schedule, or MPS) to the shop floor. Master schedulers ensure that production levels are sufficient to meet customer orders; they act as the buffer between the production shop and the external customers for one or more groups of products. To perform these functions, master schedulers rely on their factory's MRP-II (Material Resource Planning) system[2] to maintain the current production plan, along with information such as future orders, future forecasts, and material availability. The MRP system maintains the production bill of materials and facilitates the timely ordering of material required to support the planned production.

However, when changes occur in customer orders or component availability, the master schedulers must determine whether the planned production schedule is still feasible. If an infeasibility exists, they must use their knowledge of the current levels of shop activity and specific customer requirements to determine actions to minimize the effects of that infeasibility, e.g., by expediting material or modifying the schedule. In addition, master schedulers may determine staffing levels in work centers responsible for their products.

To perform these functions, master schedulers will work with several time horizons. The shortest of these typically extends a week or two into the future, and is concerned with how best to ensure that the final production plan is made. A scheduler working in this time horizon is concerned with work center capacities, the on-hand quantity of components, job sequencing and—because of its impact on the total capacity available for production—machine setup times. However, the scheduler's flexibility in dealing with capacity shortages in this time horizon can be hindered because the material required to support production most likely already has been released to the shop, which is not under the scheduler's direct control.

A second, medium time horizon extends from approximately 1 to 3 months into the future. Here, the master scheduler wants to be sure the schedule is realistic, e.g., that orders are coming in according to forecast and that expected material availability has not changed. If schedulers have sufficient advance knowledge of potential capacity problems, they retain considerable flexibility to deal with them, because they can still expedite material, rearrange production across time, and reassign personnel to different workcenters.

Finally, there is the other extreme of the time horizon, where schedulers also may assist in long-range planning—roughly six months to over a year in the future. This extended time horizon is concerned primarily with ensuring that the gross system capacity agrees with the long-range forecast of orders.

Most MRP-II systems have a "rough cut" capacity planning tool to help schedulers identify potential capacity shortages. Typically, these modules report the load on a group of user-specified resources by a group of user-specified products, usually one or more product families. However, they do not help the planner determine a *response* to any capacity shortages, e.g., by helping the scheduler identify workable alternatives to the original infeasible plan. The rationale is that an automated decision adjustment process would be impossible, undesirable, or both because of the intangibles affecting the determination of a feasible final schedule. However, the manual adjustment process is wasteful, not only of critical resources but also of the planner's time.

Schedulers need a tool to help them choose among the numerous feasible alternative schedules. Ultimately, a master scheduler's success or failure is determined by the perceived quality of those schedules. A person has been given that job because machines cannot deal with all the intangibles required to evaluate a schedule's quality. Our goal is to provide an interactive tool to guide schedulers to their "optimum" schedules, i.e., what they would like to have arrived at themselves, given enough time, patience, and cleverness in trying different approaches to resolve the infeasibilities. IRAM was designed to overcome the shortcomings of available rough cut capacity planning tools. It can help reduce waste by automatically ensuring a desirable product mix and full use of critical resources.

The required tool must be interactive and responsive to changing conditions and various intangible forces. It must work with the schedulers, allowing them to insist on minimal levels of production for each product, to prioritize products, and to control the amount of production devoted to "normal" versus "backscheduled" production. Its logic must include most standard MRP considerations. For example, lead-time offsets—i.e., the time between consumption of the critical resource and production of the final product—must be allowed to vary across each product and resource pair. It must also be able to solve the same size problems as those faced by individual schedulers. An individual product family can have between 10 and 250 different products. Typically, schedulers will want to do their simulation check with no more than 50 resources—that may represent material or capacity resources—and a time horizon of anywhere from two to six months. While this size does not allow simulating the entire factory as a unit, it does extend to the size of individual responsibility. Finally, the implementation and user interface must allow the user to have considerable control over the tool's operation.

IRAM's Mathematical Algorithm

To build such a tool, we must first discuss the mathematical algorithm it might use. The basic problem is finding the production schedule closest to the original. The main modeling issue is how to define the distance between the original schedule and a modification. In the past, there have been many different approaches to this problem.

One natural measure uses a linear objective function to minimize the value of the lost production (*v.* Meketon[3]). This economic objective has three primary advantages:

- It is natural to users.
- High-quality linear programming packages are readily available.
- It is easy to add additional features to the model without modifying the solution algorithm.

However, a linear objective has disadvantages. First, its solutions tend to include production cutbacks that vary noticeably between different products. In addition, the solution vectors are discontinuous in the value coefficients used for each product. Thus, small changes in the input parameters can lead to wild swings in proposed production schedules. This results in two practical problems:

- At any given time, the solutions cannot be gently "steered" to an acceptable solution.
- The proposed production schedule is likely to vary noticeably from iteration to iteration when this technique is used repeatedly.

These problems are compounded when the true value of a unit of lost production is largely unknown because of the intangibles, including customer satisfaction and market presence, that affect it. Thus, users might have to experiment with a range of input values for each product before they find a suitable solution. However, the long run times for linear programming algorithms that operate on problems this size are not appropriate for an interactive tool. As a result, it is difficult to see how a linear programming approach can help create the type of tool described above.

Mitchell[4] used a quadratic objective function in his hierarchical production planning model. While this tends to produce more balanced solutions, the necessary run times are even longer. Thus, such a tool cannot be considered truly interactive.

Luss and Smith, however, aware that the existing mathematical algorithms do not fit the needs of the users, developed an algorithm based on a different notion of closeness. We will now turn to a presentation of that algorithm.

The Luss and Smith Minimax Algorithm. We begin with a brief summary of the original Luss and Smith model for solving a single-period resource allocation problem. Readers are encouraged to consult the original paper or a similar paper by Tang[5] for greater specificity.

We begin with the basic notation. For each product, the original demand is denoted by d_j, the lower bounds for production by l_j (where each $l_j \geq 0$), and the relative priority of the product by α_j. The available capacity of resource i will be denoted by b_i, and the utilization matrix is denoted by $A = (a_{ij})$, where a_{ij} represents the amount of resource i used by each unit of product j. Then, a production plan $\mathbf{x} = (x_j)$ is feasible if:

$$\begin{aligned} &\sum_j A_{ij} x_j \leq b_i && \text{for all resources } i \\ &x_j \geq l_j && \text{for all products } j \end{aligned} \tag{1}$$

Luss and Smith's objective is to keep the maximum weighted percentage production cutback as small as possible, i.e., to find the production plan x for which:

$$\max_j \ \alpha_j (d_j - x_j)/d_j \tag{2}$$

is as small as possible, subject to the constraints in equation (1). While standard techniques can transform this objective into a linear function, Luss and Smith have developed a special-purpose algorithm to solve this model in orders of magnitude significantly faster than LP solvers. Recognizing that there are many production vectors **x** with the same minimax objective value, Luss and Smith make the additional requirement that the final solution should be what they called a "lexicographic" minimax solution. The nature of this type of solution will be explained in greater detail shortly.

First, we give the basic details of finding a minimax solution. It is easy to see that the problem can be transformed to one with all lower bounds $l_j = 0$. Then, for each resource i, define:

$$S_i = \sum_j a_{ij} d_j \qquad (3)$$

and

$$T_i = \sum_j a_{ij} d_j / \alpha_j$$

Thus, S_i (and T_i) are the total (weighted) requirements for resource i in the original production schedule. Notice that if $S_i \leq b_i$ for all i, we can set $x_j = d_j$ for all j and stop. Next, define:

$$k_i = (S_i - b_i)/T_i \qquad (4)$$

and

$$k = k_{i^*} = \max_i k_i$$

The individual k_i measures the extent to which resource i is "overloaded," and their maximum k gives the resource i^* that is most overloaded. Then, we define the production vector **x** by

$$x_j = d_j(1 - k/\alpha_j) \qquad (5)$$

Luss and Smith show that equations (4) and (5) give the solution to the minimax objective function if the constraints $x_j \geq 0$ are ignored. In this case, k gives the optimal minimax value, and equation (5) gives the associated production schedule. Furthermore, they show that if a value of x_j (for some product j) computed by (5) is negative, then there is an optimal solution to the original problem with $x_j = 0$. In this case, the algorithm fixes such x_js to zero, excludes these products from the problem, and repeats the calculations in (3) through (5) with the remaining products until finally all the x_j are nonnegative and an optimal minimax solution is reached.

Luss and Smith realized that additional production is possible for those products that do not use this most critical resource. Thus, their algorithm uses an iterative procedure to reach the final production plan. While these steps do not change the value of the minimax objective function, they are crucial to determining a good production plan. At each of these "lexicographic" steps, the most critical resource is determined by the minimax algorithm. The products blocked by that resource—i.e., the products that $a_{i^*j} > 0$—are eliminated from the problem. The resulting smaller problem is then solved in the same way, until the problem is feasible or all the products have been blocked.

Much of the speed of the Luss and Smith algorithm stems from update rules that can be used to simplify the lexicographic steps. Thus, most of the algorithm's work is done in the first iteration's calculation of S_i and T_i. For a discussion of the computational speed of a similar minimax algorithm as compared to other LP solvers, see Klein, Luss, and Smith.[6]

General Discussion of the Model. The speed of the Luss and Smith model makes it possible to solve interactively large-scale production planning problems. Hence, this algorithm is the most suitable for use as the basis for IRAM.

However, the Luss and Smith minimax model has disadvantages, particularly because the algorithm cannot easily be modified to add more modeling features. Furthermore, it has two serious drawbacks that

must be overcome before it can be used in IRAM:
- It solves only single-period problems.
- There is no explicit method for ensuring integer production levels.

Tang gives an algorithm to find an exact integer minimax solution. However, we choose to use a heuristic method in order to decrease the algorithm's run time.

In the next sections, we will describe the modifications made to add these features to the original algorithm.

This minimax objective function may also be somewhat less discriminating than an economic objective. Thus, these lexicographic minimax solutions may be better referred to as simply being Pareto optimal: that is, there is no way to increase production of any one product without decreasing production for another product. A successful implementation using the minimax objective requires that users know enough about the products to set correctly their lower bounds and relative priorities. The advantage of the Luss and Smith model is that its optimal solution vectors are continuous in the input parameters. This allows the users to control smoothly the algorithm's proposal to the final optimal solution.

Linear and quadratic objectives also require the user to set correctly these or similar parameters. However, the long run times for algorithms using these objectives functions mean that heuristic methods will most likely be used to determine these settings without, or with minimal, user intervention, as noted by Mitchell.[4] In contrast, an implementation of the Luss and Smith algorithm can treat these soft inputs as "control parameters" that users can easily change if they do not like a specific feature of the current solution. Over time, users will empirically learn the "responsiveness" of the solution to these changes, and hence, be able to pinpoint quickly an optimal solution for them.

IRAM's Multi-period Algorithm. During the development of IRAM, Luss and Smith,[7] working together and with Klein, developed a multiperiod algorithm using a minimax objective function that is cumulative over time. However, this algorithm requires that all resources be storable, i.e., material rather than capacity constraints. If so, it is always possible to reserve capacity available in one period to fulfill production due in a later period. Thus, there would never be any need to pre-produce. However, since IRAM must be able to simulate both material and capacity resources, this algorithm cannot be used. In addition, the multiperiod problem cannot be solved as a series of independent single-period problems because the algorithm must be able to handle lead-time offsets.

Thus, we convert the multiperiod problem into an "equivalent" single-period problem. First, we expand the product and resource lists to include a separate "artificial" product and resource for each period in the analysis. Thus, if the original problem has n products, m resources, and T time periods, this "expanded" single-period problem has nT products and mT resources. For simplicity of exposition, we will represent each of these nt products as a pair $\langle j,t \rangle$ that indicates the "real" product (between 1 and n) and time period (between 1 and T) to which this artificial product refers. Similarly, the resources in this expanded single-period problem will be referred to as pairs $\langle i,t \rangle$.

The next step in defining our "expanded" single-period problem is to describe how the multiperiod data will be used as data for the single-period problem. For the production data (product demands, lower bounds, and relative priorities), we use the natural correspondence. For example, $d_{\langle j,t \rangle}$ is the demand for product j in period t. Defining the capacity of the resources in this expanded problem is slightly more complicated. Given the original multiperiod problem capacities $c(i,t)$, we define the expanded single-period capacities $b_{\langle i,t \rangle}$ by:

$$b_{\langle i,t \rangle} = \begin{cases} c(i,t) & \text{if resource } i \text{ is not storable} \\ \sum_{s \le t} c(i,s) & \text{if resource } i \text{ is storable} \end{cases} \tag{6}$$

Similarly, we define the utilization matrix $a_{\langle i,t \rangle, \langle j,s \rangle}$ of the expanded single-period problem from the basic utilization matrix of the multiperiod problem a_{ij}, as follows (assuming, for notational simplicity, that all lead-time offsets are *0*, and that the multiperiod utilization matrix does not change over time):

$$a_{\langle i,t\rangle,\langle j,s\rangle} = \begin{cases} a_{ij} & \text{if } s < t \text{, and resource } i \text{ is storable} \\ a_{ij} & \text{if } s = t \\ 0 & \text{otherwise} \end{cases} \qquad (7)$$

Because the schedules for the corresponding multiperiod and single-period problems must both be feasible, this single-period problem is essentially equivalent to the original problem. To check this, notice that in both formulations production in period s requires the non-storable resources from period s and takes away capacity of the storable resources that could otherwise have been used for production in all periods $t \geq s$.

This single-period problem could be solved using the standard Luss and Smith algorithm. However, this does not ensure that production lost at the first (and subsequent) lexicographic steps is ever recovered, because the production goals for each product are not linked over time. While this drawback will not affect the optimal minimax value, it would make the resulting schedule largely useless. Recall that a final tool must reschedule any lost production in another time period (either earlier or later), and allow users to treat rescheduled production differently from normal production.

Thus, IRAM creates and solves this equivalent single-period problem using the standard Luss and Smith algorithm with a modified lexicographic step. The algorithm incorporates "period linking" by explicitly reloading any lost production back into the demand vector. To give users control over the amount of resources devoted to rescheduled—as opposed to normal—production, we allow them to specify that this affected production receive a different priority than "normal" production, that is, production implemented by means of a user-specified multiplicative constant. For example, the user can increase the importance of production that is backscheduled.

Specifically, if product j in time period t is blocked at the lexicographic step, we define

$$d_{\langle j,t\rangle} = d_{\langle j,t\rangle} - x_{\langle j,t\rangle} \qquad (8)$$

In this example, $d_{\langle j,t\rangle}$ represents the total "lost" production of product j at time period t. The algorithm will attempt to make up this lost production either earlier or later than time t. In general, we search for the closest time s for which $\langle j,s\rangle$ is still free, and load in the demand $d_{\langle j,t\rangle}$ with a modified priority $\alpha'_{\langle j,t\rangle}$ into the production vector for time s.

There are two approaches we can take to implement this period linking. First, we can keep track of this rescheduled production as a product completely separate from the normal production of product j in period s. This will require additional notation: the rescheduled production might be referred to as $d_{\langle j,t,s\rangle}$ tp indicate it is production of product j produced in period s to satisfy an original demand in period t. While this is notationally complicated, it is not especially difficult to implement. However, this approach slows down the lexicographic steps because the number of products is roughly the same after each iteration, with a smaller quantity demanded each time.

The second approach to increasing the algorithm's efficiency is combining the rescheduled and original production. This obviously is possible if the priority of the rescheduled production does not change. However, the users' need to be able to separate their priorities is deemed very important. As a result, we have turned to the following heuristic method of combining these two demands:

- Let an "original" demand $d_{\langle j,s\rangle}$ with priority $\alpha_{\langle j,s\rangle}$ and a "rescheduled" demand of $d_{\langle j,t\rangle}$ with priority $\alpha'_{\langle j,t\rangle}$ be "equivalent" to a total demand of $d_{\langle j,s\rangle} + d_{\langle j,t\rangle}$ with a combined priority of

$$\frac{(d_{\langle j,s\rangle} + d_{\langle j,t\rangle})}{(d_{\langle j,s\rangle}/\alpha_{\langle j,s\rangle} + d_{\langle j,t\rangle}/\alpha'_{\langle j,s\rangle})} \qquad (9)$$

This combined priority was derived to make S_i and T_i of the Luss and Smith algorithm unchanged. As a result, combining these two demands in this way usually will not affect the algorithm's results. However, if at the next iteration, the critical k value of the uncombined problem falls between the priority levels $\alpha'_{\langle j,s\rangle}$ and $\alpha'_{\langle j,t\rangle}$, then one of the "separate" products would have its demand set to zero. In this case, it is demonstrable that the production assigned in the two methods is different.

Our computational experience indicates that using this heuristic—as opposed to the algorithm where the demands are left separate—results in an approximately 25 percent decrease in the algorithm's run time, while the potential error seems to be minor. As a result, IRAM is able to generate high-quality solutions to multiperiod models with the original philosophy of the Luss and Smith algorithm, but with little or no extra computational overhead.

IRAM's Integerization Heuristic. The other major modification we need in the basic Luss and Smith algorithm ensures integer production levels. At first it might seem that this might not be necessary. Because IRAM checks only the rough cut feasibility of the schedule, it might seem that simply rounding down the computed production schedule to the nearest integer might be sufficiently accurate. However, if the production plan to be checked has small production levels, then simple rounding may leave substantial amounts of unused resources. As mentioned above, Tang's exact integerization technique was deemed too slow for our purposes, for it typically requires many iterations of the basic algorithm. In this section, we will describe the heuristic method IRAM uses.

First, we will run our multiperiod modification of the Luss and Smith algorithm through all its lexicographic steps, rounding down the resulting lexicographic minimax solution to the nearest integer. The main advantage of this method—as opposed to applying an integerization procedure after each lexicographic step—is that we can use the fast lexicographic updates.

Now we want a procedure for determining a good use for the resources that have been freed by rounding down the lexicographic minimax solution. Thus, for each product, we compute the total fraction of satisfied demand over the entire time horizon $(\sum_t x_{\langle j,t\rangle}/\sum_t d_{\langle j,t\rangle})$. These values are sorted in increasing order, and the products are processed in that order, i.e., with those products experiencing the largest total cut handled first. This method is "greedy" in that products are processed in this order, with the largest possible production increases being made at each step, irrespective of the possible effect on other products.

For each product, we make three passes.

1. We check to see whether the production levels for our product can be increased closer to the original demand for it in any time period.
2. If so, production in each such time period is increased as much as possible to meet the original product demand. By doing so, we may have eliminated the need for overproduction in other time periods; hence, a second pass eliminates any such overproduction.
3. This pass determines whether any remaining shortfall between the original total demand for the product and its current total production can be made up during any time period. If it can, we increase the production as much as possible in these periods.

Clearly, this integerization method is crude. In other applications, more rigorous approaches may be required. For example, even after our procedure, and although the greedy procedure we have described makes it unlikely such additional passes will result in substantial changes, additional passes through the entire product list may still be necessary to make sure no leftover resources could be used by any other product. In addition, in the first pass for each product, it may be better to increase each period's production up to only one more than the continuous lexicographic minimax solution rather than all the way up to the original product demand. Other possible techniques may include marginal allocation schemes in which $x_{\langle j,t\rangle}$ is increased for only a single $\langle j,t \rangle$ at a time, or approaches similar to that of Tang.

In our experience, however, this method works well without adding significantly to the algorithm's overall running time. But as we have stressed, the final run time is of paramount importance to the tool's final acceptability.

Implementation and User Interface

We next describe IRAM's implementation and user interface, concentrating on the interactions between the user's needs and the algorithm's requirements integrated into a final tool. Users require a tool that can quickly solve planning problems using the same logic as their MRP system, allowing them control of the tool's operation to make it responsive to the intangibles that are part of choosing the best production schedule.

The algorithm chosen as the basis for the tool's operation requires user input to steer it to the best solution. A critical decision at this point is the appropriate computer environment and user interface that will be the best bridge between the mathematics and the users.

It was determined that a PC would be used for the development of IRAM. A PC has substantial advantages. First, it is an "offline" system. Any changes the master scheduler simulates do not affect the production database. Second, a PC application typically can be more interactive than the mainframe MRP system. Finally, PCs are increasingly common in factory environments. Many master schedulers already use "home-grown" spreadsheets to help determine schedule feasibility.

One drawback of PC-based tools is the difficulty of inputting new data. While IRAM can be used without an interface to the local MRP system, such an interface is required as a practical matter to ensure that the user can quickly obtain and process the required data. IRAM uses a generic input file format that makes it easy to develop such an interface from most MRP systems.

PCs also can provide development problems because they have limited memory resources and their computational speed is inferior to a mainframe's. Thus, while the Luss and Smith algorithm has only moderate memory requirements, IRAM's implementation required resourcefulness to prevent the PC's limitations from significantly affecting its performance.

With respect to the IRAM user interface itself, the goal was to provide an environment where users could critically evaluate any resource shortages as well as IRAM's recommended schedule. This is a common problem for developers of many interactive support tools: they need to efficiently code the main part of the algorithm, but still develop an interface that is both easy to use and "bullet proof," i.e., one that captures user errors such as typos. IRAM offers a paradigm for a powerful interface that is easy to add to many other support tools.

Our interface combines the resource allocation "engine" with a standard spreadsheet program, currently either Lotus Development Corporation's 1-2-3™ or Symphony™. The allocation engine implements the modified Luss and Smith algorithm and is coded in the FORTRAN programming language with standard input/output (I/O) routines. The spreadsheet has macros to automate the repetitive tasks—such as generating the engine's input file and reading in the engine's output into the appropriate worksheet ranges—required to use the engine. In addition, additional spreadsheet macros simplify other tasks for the user, including moving around the worksheet, running the MRP system interface program, and running various types of output analysis.

This arrangement offers significant advantages to the tool developer. The main engine driving the tool can be coded using a simple flat file interface, thereby reducing the amount of input and output analysis the engine is required to perform. Such analysis becomes the spreadsheet's responsibility. The developer can also rely on the spreadsheet for most of the bullet-proof I/O routines. Finally, the modularity of the entire package is improved because the interface is almost totally separate from the tool's engine.

There are advantages to the user as well. Users interact with the tool solely through the spreadsheet, which presumably is natural and familiar to them. This can reduce the user's total learning time because of the ease of use of spreadsheet programs. In addition, the user has the full power of the spreadsheet for output analysis, including easy data entry and modification, printed reports, graphics, and database management.

The IRAM worksheet has a menu tree defined with macros. The user can use these menus to choose to view input and output data, including the MPS (the original product demands), the FPS (IRAM's proposed Feasible Production Schedule), and the BOR (the Bill of Resources summarizing the product/resource utilization matrix). In addition, other menu choices run external programs, including the main IRAM engine, the MRP interface program, and a program presenting utilization details for a specific resource. Finally, the user can run spreadsheet-based analysis routines that include graphs of a specific product's schedules and resource utilizations, "exception" reports for back-scheduled items, and comparisons of previously saved proposals. Some of the IRAM menu options are shown in Figure 1.

A final example shows the additional power and flexibility of the spreadsheet interface. In early discussions of the tool, it became clear that users wanted to be able to input the lower bounds in two ways: as absolute quantities, and as a percentage of the original schedule. Similarly, they wished to view the resource loads as absolute quantities and as percentages of the available capacity. In a standard interface, these capabilities would be added with additional input variables that indicate whether input or output data represented absolute or fractional quantities. For IRAM, however, the main application program need not be changed; instead, the data in the worksheet ranges can be changed from absolute quantities to formulas. Indeed, if the formulas are simple enough, the users may be able to create them. Thus, substantial flexibility in the input routines and output analysis can be arranged without modifying the main application program.

View input data

- MPS
- Lower bounds
- Weights
- BOR
- Capacity

View output data

- FPS
- Resource utilization
- Cumulative view
- Save current FPS

Run external programs

- Allocation engine
- MRP interface
- Detailed resource allocation

Miscellaneous analysis

- FPS vs MPS (difference, cumulative, or percent)
- Current FPS vs saved FPS
- Summary reports (to printer)
- Exception reports of backschedule
- Graphs (of products, resource)

Figure 1. IRAM menu options.

The techniques used to link an external application to a commercial spreadsheet are generic and could be used with many other PC-based support tools. Of course, some applications could not use this type of interface because they do not fit the spreadsheet paradigm. For example, applications that are totally graphics oriented or that are based on expert systems probably would not be appropriate. In addition, the relative slowness and memory limitations of a PC compared to mainframe computers might rule out other applications. However, we feel that many interactive "number crunching" applications could use such an interface.

Conclusion

One of the reasons for interest in IRAM is the way the various phases of final tool development interacted. In particular, the user's requirements for a support tool (rather than a "black box") led us to our choice of the objective function, which requires fundamental levels of user input. This, in turn, imposed requirements on the tool's user interface so users could critically evaluate the quality of the proposed production plan and control the types of solutions the tool generates. These considerations influenced the choice of computer used for the tool, and the types of heuristic methods that could be used to add specific features to the basic Luss and Smith model. While we have presented the story of IRAM's development in a "straight line" from the user's needs to the development of the tool's user interface, the different parts of the development process actually came together simultaneously, with the final tool representing a true example of research and development.

As optimization analysts move increasingly toward trying to help solve problems for which there are vague or varying criteria for judging the quality of a solution, the need will increase for support tools to interact with the end-users' knowledge and experience with the problem. For these tools, the lessons learned by studying IRAM and similar projects will become increasingly valuable.

Acknowledgments

I would like to express my thanks to some of the many of the people who helped in the development of IRAM. The previous technical work by many people (including H. Luss, J. C. Mitchell, and D. R. Smith) helped me formulate the heuristics used by IRAM, and S. McQuin and P. Warren helped provide the user's perspective. In addition, many people (including anonymous referees) help improved the readability of the manuscript. Finally, M. Meketon developed a spreadsheet interface for a similar tool that served as the initial model for IRAM's interface.

References

1. H. Luss and D. R. Smith , "Resource Allocation Among Competing Activities: A Lexicographic Minimax Approach," *Operations Research Letters*, Vol. 5, No. 5, November 1986, pp. 227-231.
2. J. Orlicky, *Material Resource Planning*, McGraw-Hill, New York, 1975.
3. M. S. Meketon, "The Component and Capacity Constrained Plan (C^3P) Program," presented at the National Bureau of Standards Conference on *Real Time Optimization in Automated Manufacturing Facilities*, Gaithersburg, Maryland, January 21, 1986.
4. J. C. Mitchell, "A Hierarchical Production Planning Model for Allocating Materials in Short Supply," presented at the Joint National Meeting of ORSA/TIMS, Atlanta, Georgia, November 4-6, 1985.
5. C. S. Tang, "A Max-min Allocation Problem: Its Solutions and Applications," *Operations Research*, Vol. 36, No. 2, March/April 1988, pp. 359-367.
6. H. Luss and D. R. Smith, "Multiperiod Resource Allocation: A Lexicographic Minimax Approach," presented at the Joint National Meeting of ORSA/TIMS, St. Louis, Missouri, October 25-27, 1987.
7. H. Luss and D. R. Smith, "Multiperiod Allocation of Limited Resources: A Minimax Approach," *Naval Research Logistics Quarterly*, Vol. 35, No. 4, August 1988, pp. 493-501.

(Manuscript received December 23, 1988)