

**THE R.J. REYNOLDS TOBACCO COMPANY
AWARD FOR EXCELLENCE
IN TEACHING, RESEARCH, AND EXTENSION
DISTINGUISHED LECTURE SERIES**

**Monkey with a Typewriter:
Solving Problems in Industrial Engineering**

Dr. Thom J. Hodgson
Distinguished University Professor
of Industrial Engineering
and James T. Ryan Professor of Industrial Engineering
and Furniture Manufacturing
Director of the Integrated Manufacturing Systems
Engineering Institute
North Carolina State University



**College of Engineering
North Carolina State University
Raleigh, North Carolina**

Based on the Lecture Presented October 20, 2005

TWENTY-FIRST RECIPIENT

THOM J. HODGSON

**THE R.J. REYNOLDS TOBACCO COMPANY
AWARD FOR EXCELLENCE
IN TEACHING, RESEARCH, AND EXTENSION**



Thom J. Hodgson

An internationally renowned scholar in industrial engineering and operations research, Dr. Thom J. Hodgson is a dedicated engineering researcher and educator. A member of the National Academy of Engineering, he is one of the pioneers in the study of the design and analysis of supply chains.

Dr. Hodgson, who joined the College of Engineering faculty in 1983, is the founding director of the Integrated Manufacturing Systems Engineering Institute (IMSEI), which provides multidisciplinary graduate-level education and practical training opportunities in the theory and practice of integrated manufacturing systems engineering.

A Distinguished University Professor of Industrial Engineering and the James T. Ryan Professor of Industrial Engineering and Furniture Manufacturing, he has received much recognition for his professional achievements. He was one of the 125 inaugural Fellows of the Institute for Operations Research and the Management Sciences (INFORMS). He is a Fellow of the Institute of Industrial Engineers (IIE), serving twice as vice president of IIE.

In 2004 he received the Frank and Lillian Gilbreth Industrial Engineering Award. Given by the IIE, the award recognizes individuals who have distinguished themselves “through contributions to the welfare of mankind in the field of industrial engineering.” In 2003 he received the Albert G. Holzman Distinguished Educator Award from IIE for his significant contributions to the profession through research, publication, extension, administration, and teaching innovation in the academic environment. Recognized for teaching excellence, he has also received twice the C.A. Anderson Outstanding Faculty Award, which is a student-initiated award, and an Alumni Distinguished Graduate Professorship.

He has supervised 31 PhD candidates and more than 40 master's degree students.

Dr. Hodgson is the author or co-author of more than 70 journal articles and book chapters. His areas of research interest are scheduling, logistics, production and inventory control, manufacturing systems, and applied military operations research. He has served as associate editor, departmental editor and editor-in-chief of *IIE Transactions*.

Dr. Hodgson received his bachelor's degree in science engineering in 1961, his MBA in quantitative methods in 1965, and his PhD in industrial engineering in 1970, all from the University of Michigan.

He served as head of the Department of Industrial Engineering at NC State from 1983 to 1990, and he was director of the Division of Design and Manufacturing Systems at the National Science Foundation from 1991 to 1993. Prior to joining the faculty at NC State, he was a professor of industrial and systems engineering at the University of Florida from 1970 to 1983 and an operations research analyst at Ford Motor Company from 1966 to 1970. He served as an officer in the US Army from 1961 to 1963, and was a member of the US Army Science Board from 1994 to 2000, earning the Certificate of Appreciation for Patriotic Civilian Service.

An advisor and mentor to his many students, he treats his graduate students as colleagues in research and publications, and he demonstrates his trademark wit through a unique method of motivation for his undergraduates by promising an "A" to any student who can beat him at handball, a feat that has yet to happen.

Contents

Abstract.....	7
Introduction.....	7
Example 1: Packfab: Cutting Markers from Fabric for Furniture Manufacturing.....	9
Example 2: A Material Allocation Scheme for Optical Fiber Cable Manufacturing.....	11
Example 3: Dry-or-Buy Decision Support for Dry Kiln Scheduling in Furniture Production	18
Example 4: Satisfying Due Dates in Large Multi-factory Supply Chains	27

Abstract

There is an old adage that says, if you give a monkey a typewriter and let it type long enough, it will eventually produce the Encyclopedia Britannica. Well, we've got a reasonably smart monkey (your basic industrial engineer) and a really fast typewriter (a multi-gigahertz PC). This seminar is about how we can use analysis and the concept of randomization to solve real combinatoric problems. We review a number of industrial problems and their real solutions. A key issue is to keep the technology low enough that fundamental analysis of the problem and good coding skills will be sufficient for success. In what follows we look at a series of problems that have yielded to the monkey (in this case your standard industrial engineer) and the typewriter (your standard PC, a very fast typewriter).

Introduction

At 30 years old, I had little perspective on my professional career. After reaching 65 years, you start to understand the driving forces in your life and the effect they have had on your work. My first real job as an industrial engineer came before I had any degrees in industrial engineering. I discovered industrial engineering in the middle of pursuing an MBA at the University of Michigan. By the time I completed the MBA, I had taken a bunch of additional courses, which were the equivalent of a Master's degree in IE. I took a job in the Operations Research group at the Transmission and Chassis (T&C) Division of Ford Motor Company in Livonia, Michigan.

The T&C Division was what my military friends would call a "target rich" environment. There were all kinds of wonderful operational and engineering problems to work on. In addition, Ford had just purchased one of the very first GE time-sharing computers. Real-time computing was suddenly available to anybody who had a Teletype machine. I was already a decent computer programmer. I was at the right place, at the right time, with the right analytic and computational tools.

This was a very pragmatic environment. The emphasis was on

solving problems and creating satisfied customers in the division. The projects run/span the gamut or represent a range of industrial engineering applications: a computerized system for the financial analysis of projects, several different on-line production scheduling systems, automated testing of transmissions, optimal gear design, reverse engineering standard transmissions to determine the original design life criteria, simulating the process of shifting in an automatic transmission to test virtually various materials to be used in the clutch plates, and teaching more than 100 engineers how to write computer programs in Basic.

This early focus on solving the operational problem and then implementing the result has driven my work for 40 years. The limiting factor is that many of the problems that we all work on are computationally very intense. We try to find mathematically optimal solutions to operational problems with minimum computational effort. To get optimal, or near optimal, solutions to many of the operational problems that we work on in a finite length of time, sometimes may not be possible. In fact, if the objective is to develop the mathematical model of the operation or system to the level of detail that it can actually be implemented in the real world, you are almost assured that the solution is going to be extremely difficult.

Another driving factor is the “time value for analysis.” If the solution to the problem is not obtained in a timely manner, it may be too late for implementation in the real environment. Many operational problems have very limited time windows for analysis. Most operational problems can be solved using heuristics (specialized procedures that obtain solutions of varying quality). Many times these heuristics can be analyzed for average or worst case performance. However, worst-case performance may not be good enough for the real problem. So we need to find a way to enhance the solution to near optimal within the time available for solution. Enter the monkey with a typewriter. In this case the monkey is an industrial engineer, and the typewriter is a personal computer.

Now we all know that computers are fast. But it turns out that how they are programmed makes a big difference in speed. For example, consider a computer multiplying two numbers together. Computers store numbers as either “real” (floating point) or “integer.” If the computer

takes ‘x’ time units to multiply 2 ‘integer’ numbers, it takes ‘10x’ time units to multiply 2 real numbers. If it takes ‘x’ time units to add 2 ‘integer’ numbers, it takes ‘5x’ time units to add 2 real numbers. This is because ‘real’ and ‘integer’ numbers are represented in different ways in the computer. The point is that if time is of the essence, how you program your algorithm (both data type and data structure) can make a big difference in critical performance.

What follows are four examples of a monkey with a typewriter. In each case systems were built for the companies or the military.

Example 1: Packfab: Cutting Markers from Fabric for Furniture Manufacturing with Tom Culbreth, Tim Gurganus, Russell King

“Packfab” is a computer program designed to set up markers. It is a classic example of a monkey with a typewriter. The problem is simply to place patterns that define the individual fabric parts for an upholstered chair, couch, loveseat, etc., on a bolt of fabric so that the pieces can be cut and sewn together. The issue that tends to complicate the process is that some pieces can only be cut from certain places on the fabric. This is because stripes and plaids must match up when the pieces are sewn together, or the big rose must be in the middle of the pillow, etc. The places where these pieces can be put are called match points and comprise about one-half of the pieces. Ladies with long tenure were doing that job. They were very good. However, they were retiring with no replacements.

To make a long story short, a reasonable way to make up a marker is to place the pieces with match points first, placing each piece as far to the left (Figure 1) as possible. Then place the non-match point pieces around them as far to the left as possible. In Figure 1, the match point pieces have dashed lines attaching to the match points. The problem is that this implies that the pieces have been laid out in some order, and the order of the layout affects the solution. However, with a monkey who is really, really fast, lots of layouts can be tried by randomly ordering the

pieces before each layout. By dumb luck, a really good solution might be obtained. The key to making the monkey really, really fast is to make sure that he uses a data structure for ‘organizing’ the pieces that is very efficient.

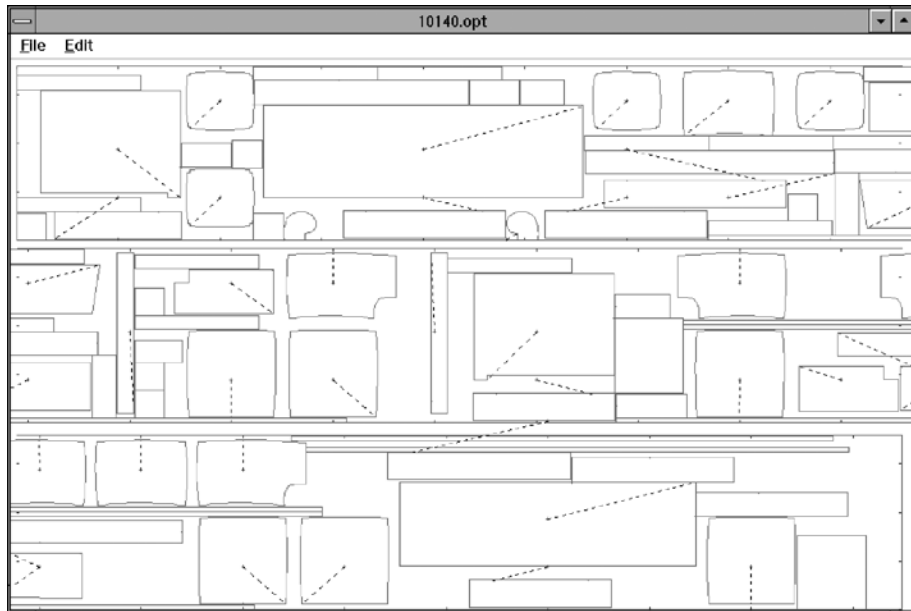


Fig. 1: Example marker for a couch.

To test the monkey we obtained a bunch of actual layouts used by industry and put the monkey to work to see how he would do. Well, the monkey didn’t always do all that well. In many cases the monkey was much better. But in some other cases the little old ladies beat the monkey. However, after further examination of the data, it was determined that the little old ladies were cheating! They used slight offsets from the match points in order to squeeze pieces in.

In the final analysis, we taught the monkey to cheat. The monkey program was sold to a small company (Cutting Edge) founded by some MIT graduates that made automated fabric cutters. They did a wonderful job of integrating the monkey into their software. The largest manufacturer of cutters in the country (Gerber) bought out Cutting Edge. So the monkey program is now the industry standard.

Example 2: A Material Allocation Scheme for Optical Fiber Cable Manufacturing with Steve Jackson, Peijun Qu, Reuben Cannon

The manufacture of fiber optic cable involves a series of operations that combine continuous lengths of color-coded optical fiber into finished cable. The job was to devise a method that allocates the optical fiber in inventory (both pre-colored and natural, or non-colored) to orders for finished cable of customer specifications and lengths. Cable orders include specific optical fiber attributes including color, type, and attenuation levels. A fiber remnant may remain on the spool after it is used to fill an order. If the remnant is short enough (\leq Maximum Scrap Quantity (MSQ)), the fiber is discarded as scrap. If the remnant is longer than the Minimum Re-use Quantity ($MRQ \geq MSQ$), it is returned to inventory for future use. An allocation scheme should (1) minimize the scrap remnants and (2) avoid all fiber remnants in the range MSQ to MRQ.

Product Structure: A finished cable consists of a number of colored buffer tubes each of which contains a number of colored optical fibers. The number of tubes varies as do the quantity of fibers within a tube. Figure 1 shows a cross sectional schematic of a finished cable. In this figure one of the tubes contains 6 fibers while the others contain 12 pieces. It is possible to have dual layer cables that have another layer (not shown) where a second circle of buffer tubes surrounds the inner layer. To allow for easy in-field identification, colors of tubes are not repeated in the cable and colors of fiber are not repeated within a tube.

Fiber Coloring: Fiber is received from manufacturers on spools in a “natural” uncolored state. When fiber is colored, the entire spool is processed. Fiber can be stored as natural fiber, or colored as occurs when it is returned to stock partially used. When allocating fiber from stock, it is desirable to use the colored stock first.

Each individual fiber spool is classified by attributes of the fiber. Examples of attributes utilized in classifying and matching demand are unique part number, length of fiber on the spool, color, fiber type, and attenuation grade.

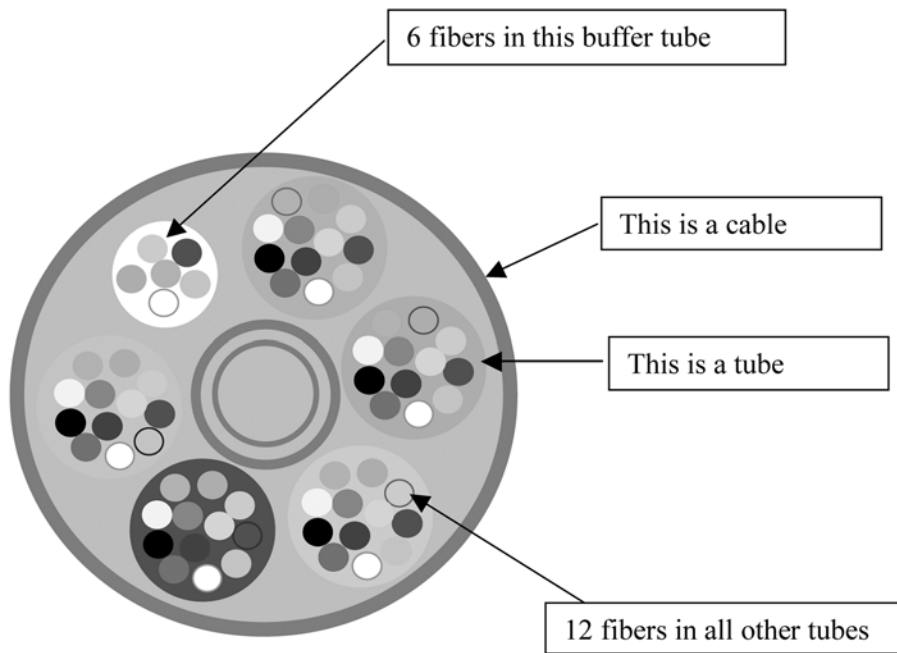


Fig. 1 Schematic of finished cable.

Buffering: Buffering is a process in which combinations of colored fibers are placed together in an extruded plastic “buffer tube.” The buffer tube is color coded for easy identification in the field. The fibers are fed from the spools and the buffer tube is extruded around them.

Stranding: After buffering is completed, several buffer tubes can be grouped together into a “stranded core” through a process called “stranding.” Stranding wraps buffer tubes around the central member and adds various materials under and over the buffer tubes. Multiple stranding operations may be needed depending on the number of buffer tubes that are needed to make the cable. If a second stranding operation is needed, an additional layer of buffer tubes is wrapped around the stranded core from the first pass, resulting in a “dual layer” stranded core. Following the stranding process the jacketing and packaging steps complete the cable.

Scheduling, grouping, and allocation of fiber: At the buffering operation, spools of optical fiber are placed at the upstream end of the

process. Fibers from all of the spools are combined and are covered with a colored extruded plastic buffer tube. This plastic tube’s color can be changed during the process without changing the spools and in this way a portion of the setup is avoided. This is known as tube grouping. In-process tube-color changes incur a small transition cost (loss of fiber), however. Plant operating policies require that tube grouping take place only within a single order where all tubes in the group have the same number of fibers and are on the same layer. In order to increase the number of unique colors in a cable, stripes may be added to the tubes. Mixing of plain and striped tubes in one group is not allowed.

In the present case, the plant schedulers predetermine a schedule so the production sequence of cable orders being produced at a buffering station is fixed. Each time the fiber spools are changed a setup cost is incurred. This setup includes loss of material and time that is typically greater than a tube color-change loss mentioned in tube grouping above. If there is sufficient fiber remaining on the spool (i.e., greater than the MRQ), it is returned to inventory where its length is confirmed and the inventory updated. If the length of fiber remaining on the spool is below the MSQ threshold, the fiber is discarded. Spools left with fiber lengths between the maximum scrap and minimum threshold values are to be avoided since the fiber is too long to discard but is not of sufficient length to fill future orders.

Since the optical fiber is the high cost material in the cable, fiber spools from inventory should be allocated to the buffering operations so that scrap is minimized. The allocation of these spools so that scrap is minimized is the objective.

Objectives: Given a candidate set of fiber spools of known type, length, color, quality (attenuation grade), and manufacturer restrictions, the objectives are to allocate the optical fiber from this candidate set to a specified customer demand while

1. minimizing fiber scrap,
2. selecting maximum allowable attenuation value,
3. meeting manufacturer and manufactured location and other order criteria, and
4. preferentially selecting pre-colored fiber.

Objectives 2 and 3 are easily performed by pre-selecting candidate reels. Objective 4 is intended to help reduce the relatively large percentage of colored reels in the inventory. The fiber inventory was on the order of 12,000 reels, both uncolored and colored. Over 50% of the reels were partially used and colored. There are 4 major types of fiber and 3 different manufacturers. Approximately one-half of the inventory is in one major type of fiber which is separated into 4 quality levels. The result is that the range of problems runs from having approximately 6,000 reels available down to problems having just a few hundred reels available for the schedule.

Formulation as a Mathematical Program: In formulating the present problem, one is immediately hit with the fact that the problem is fundamentally quadratic in nature. The decisions are of the form: which reels (of the various required colors) should be used, *and* how many tubes in the order should they cover? The number of decision variables for the problem at hand can be extremely large. Because of this and the requirement that solutions needed to be generated in just a few minutes, we realized that a heuristic approach to the problem was appropriate.

The key to our approach was to specify randomly the tube grouping prior to allocation of fiber reels. As will be seen shortly once tubes are grouped, the allocation of fiber reels can be accomplished easily and efficiently. However, due to the quadratic structure of the mathematical formulation, we were unable to find a constructive means by which to determine the set of breakpoints between orders that would result in a minimum scrap allocation. Since the allocation process was so efficient, it was clear that considering a relatively large number of breakpoint sets was feasible. Due to the lack of any other insight, specifying the breakpoints randomly appeared to be the most reasonable approach (i.e., a monkey with a typewriter).

Randomization of the tube groupings is accomplished in a straightforward manner. There are a finite number of points that determine the end of a group (i.e., at the end of an order). For each of the iterations, a potential tube group end is given a 0.5 probability of being used. This is implemented by simply generating a random number [0-1] for each

potential point and making it an end if the random number is less than 0.5.

As noted, given a tube grouping, allocation of reels can be done efficiently. The process of random tube grouping is repeated 1000 times or more. Each time the reels are allocated to the tube grouping. The best solution is saved for implementation.

Now, it is appropriate to consider the reel allocation process. Let

NR = the number of reels;

NG = the number of tube groups;

$X_{ij} = 1$, if reel i is allocated to tube group j ;

$X_{ij} = 0$, otherwise;

S_j = the set of reels that can be allocated to tube group j ; and

SC_{ij} = the scrap resulting from allocating reel i to tube group j .

The integer linear program (ILP) describing the reel allocation problem, given a tube grouping, is

$$\begin{aligned} & \text{Minimize } \sum_{i=1}^{NR} \sum_{j=1}^{NG} SC_{ij} X_{ij} \\ & \text{s.t. } \sum_{i \in S_j} X_{ij} = 1, \quad \forall j \\ & \quad X_{ij} = 0,1 \end{aligned}$$

For reels whose length is less than a particular tube group length, the associated decision variables (X_{ij}) are eliminated. Since there are NG constraints, and the constraints are separated by tube group, any basic solution to the LP relaxation of the problem is integer, and the allocation problem is easily solved. In general there will be a number of optimal allocations. In fact, a simple greedy procedure can be used to perform the allocation optimally.

It is necessary to take into account the management desire to minimize the colored reel inventory. This is done by first considering only colored reels whose scrap length for a particular tube group is less

than the MSQ. Next, priority is given to colored reels whose scrap length for a particular tube group is greater than the MRQ (generating minimum scrap). Then, priority is given to uncolored reels whose scrap length is less than the MSQ. Finally, priority is given to uncolored reels whose scrap length is greater than the MRQ (generating no scrap). The allocation procedure is extremely fast, thus enabling multiple randomizations of the tube groups. A general flow chart for the randomization and allocation are given below.

Program Flow: Figure 2 provides a high level view of program data flow. The data manipulation component handles I/O and user interface tasks while the optimization component makes the actual fiber reel assignments. The process is repeated for each of the fiber types in the current demand. Once all types are completed, a report file is written. The report is used to update the plant's inventory and assign spools to orders.

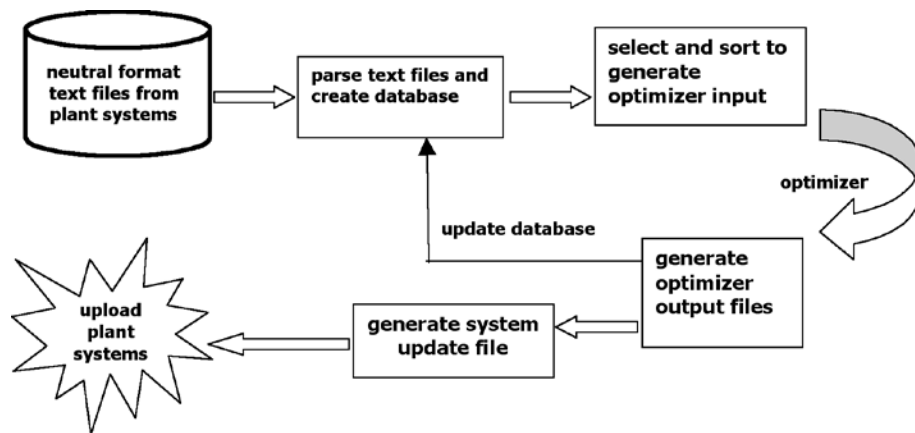


Fig. 2 Program flow.

Solution Evaluation Considerations: As noted earlier, the objective is to minimize the total fiber scrap. However, due to the mixed nature of the solutions produced, it was found that an appropriate evaluation of a set of solutions was to compute the average scrap per reel actually discarded. In other words, if a reel is returned to inventory (i.e., the remaining fiber length exceeds the MRQ and produces no scrap), its contribution is ignored. Total scrap for a solution is divided by the number

of reels not returned to inventory (i.e., the number scrapped). This tends to eliminate from consideration solutions with little total scrap due to the small number of reels scrapped. The result is that over time the number of reels in colored inventory tends to be minimized.

Performance of the Procedure: To find the true optimal solution, it would be necessary to evaluate (at least implicitly) all possible tube group combinations and fiber allocations for each order. For a normal scheduling run the combinatorics of such an effort are well beyond computational reality. However, one can gain a certain amount of confidence from observing the scheduler in action. Normally, the scheduler is run for something like 1000 iterations (controlled by the user). This takes less than one minute of CPU time. The program is set up to output to the screen whenever an improved (the monkey's best shot to date) solution is obtained. Solutions worse than the best obtained so far are ignored. A typical run might have 3-4 improvements in the first 10-15 iterations (tube groupings). There might be 2-3 improvements from iteration 10 to iteration 500. Seldom would more than one improvement occur between iteration 500 and iteration 1000. In addition, the amount of improvement for new solutions tends to decay significantly as the number of iterations goes up. The point is that there are a large number of tube groupings that result in near optimal reel allocations for a large-scale problem of this structure. It would appear empirically that 1000 iterations are sufficient to find a near optimal solution in this case.

During program development and testing, eight data sets, which consisted of multiple demand and inventory files, were provided by the company. A combination of twenty-three sets of demand and inventory could be used for testing. Exhaustive testing of the procedure on real data was critical to identifying the many special considerations necessary to the manufacturing process.

A Company Perspective: As a result of implementation of the fiber allocation system, there have been a number of direct impacts on plant operations. The standardization of the fiber assignment logic leads to more consistent allocation of pieces, which in turn minimizes operational variance and allows for better performance tracking supporting continuous

improvement. The standardization also allows for simplified training on the fiber selection process while enabling schedulers to have a larger scale view to evaluate the impact of selection criteria modifications.

We now can evaluate thousands of possible solutions while reducing the overall time and effort involved, thus fulfilling two major business initiatives in that we are able to do more work but with less effort. The fiber allocation system allows plant operations management to focus on scheduling input at a higher level. Schedulers now have the opportunity to utilize more complicated logic to find improved processing solutions without a significant investment of time to evaluate the impact on fiber scrap. In short, multiple high level scheduling scenarios can be tested in a fraction of the time with realistic scrap impact used to evaluate the benefits of the different options available.

The fiber allocation system also allows schedulers to take a broader look at the fiber inventory by evaluating placement of fiber throughout the supply chain by simply expanding or limiting the inventory available to the optimization program. The program has achieved its goals of minimizing scrap and colored inventory while adding the benefit of efficiency without compromising flexibility. The monkey triumphed again.

Example 3: Dry-or-Buy Decision Support for Dry Kiln Scheduling in Furniture Production **with Arman Yaghubian, Jeff Joines**

In order to produce solid wood furniture, a manufacturer requires dried lumber. Dried lumber has better properties than green lumber and is therefore used even though it is much more expensive. Properly dried lumber is dimensionally more stable (i.e., shrinkage, checking, splitting, and warping are reduced or eliminated) which allows the lumber to be precisely cut. Dried lumber prevents biological degradation due to fungi infection and reduces the chance of attack by insects as well as increases strength and nail-holding ability. A manufacturer usually purchases raw (green) lumber from hardwood lumber producers and brokers because of the cost of buying dried lumber. One of the main tasks for a furniture

manufacturer, prior to release of the lumber into the factory, is to dry the lumber in kilns. Lumber drying is a time consuming, energy-expensive process taking from a few days to months, depending on species and thickness. Scheduling the lumber drying kilns is an important issue for the furniture manufacturing industry.

Solid wood furniture plants have demands for lumber to be used in the manufacturing process. Each demand has associated with it thickness, species, quantity, buying cost per board foot, and due date. It is important to meet all due dates since failure to do so could result in lost production at the plant. Each kiln charge (species/thickness combination) has a known drying cost and drying time for each job/kiln combination (i.e., kilns may have different drying technologies). Each kiln has a known capacity. At the time of scheduling, some kilns may be already committed to a charge and, thus, may not be available for scheduling until some future known time. The objective of scheduling kiln charges of lumber is to deliver all the demands to the factory by the specified due dates.

Sometimes, meeting all due dates may not be possible based on current demands, kiln availability, and/or lumber drying times. To avoid lost production, the kiln scheduler may find it necessary to buy commercially dried lumber on the open market (i.e., out-source some of the lumber). Generally, buying dried lumber is considerably more expensive than buying and drying green lumber. It is important to determine the most efficient set of “dry-or-buy” decisions. One way to do this is to employ an iterative “what-if” cost analysis using a computer-based decision support system. Using associated drying and buying costs, a reasonable drying schedule can be developed by identifying certain jobs to be out-sourced and then rescheduling using the “original” schedule as a starting point. This process can be repeated until the user has obtained the “best” combination of job tardiness and cost. A drawback of this procedure, especially for large problems, is the difficulty in deciding which jobs to out-source. This approach turns out to be just a simple “trial-and-error” procedure. The approach we took was to automate the cost analysis by developing an integer-programming model of the dry-or-buy problem. In this case, the objective is to minimize the total drying-buying cost while meeting all due dates.

We assumed any number of species/thickness combinations, multiple kiln technologies and sizes, and limited kiln availability. The objective was to minimize the total drying and buying cost, while satisfying all due dates. An integer-programming model was developed. Since commercial integer-programming packages do not perform well on the problem, a randomized heuristic was developed.

The Model: In the mathematical model, it is assumed that (1) a kiln processes only one species/thickness of lumber at a time, (2) raw lumber in sufficient quantity is available as needed for each species/thickness during the scheduling period, (3) no preemption of jobs is allowed, (4) kilns are always available after the given availability time, and (5) jobs assigned to kilns are processed in due-date order. Assumption (1) is not limiting because furniture manufacturers do not mix thickness and/or species in drying kilns. Assumption (3) is not limiting since preemption would disrupt the drying process causing the lumber to degrade. Since the schedule can be regenerated if a kiln breaks down, assumption (4) is not limiting. Assumption (5) is a property of an optimal solution since for one facility, maximum lateness is minimized by processing jobs in due-date order. The problem is really one of allocating jobs to kilns. This allows the problem to be efficiently stated as an integer-program. A job may be processed using several kilns and/or kiln charges. All kiln charges are associated with a particular job, and the due date is satisfied only if the last charge processed for the job is completed by the due date.

The following notation is used:

- n = the number of demands for a specific species/thickness (job),
- m = the number of kilns,
- p_{jk} = processing time of job j in kiln k ,
- Q_k = capacity of kiln k (in board feet),
- a_k = time of availability of kiln k ,
- D_j = demand of job j (in board feet),
- d_j = due date of job j ,

- c_j^b = buying cost of already kiln dried lumber per board feet for job j ,
- c_j^g = buying cost of green lumber per board feet for job j ,
- c_{jk}^d = one charge drying cost for job j in kiln k ,

Decision variables:

- x_{jk} = the number of loads assigned to kiln k for job j ,
- z_j = amount of job j bought from outside supplier.

The integer linear program is stated as follows:

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^n c_j^b z_j + \sum_{j=1}^n \sum_{k=1}^m c_{jk}^d x_{jk} + \sum_{j=1}^n (D_j - z_j) c_j^g \\ \text{Subject to} \quad & \sum_{j=1}^n p_{jk} \cdot x_{jk} - d_j + a_k \leq 0 \quad J = 1, \dots, n; k = 1, \dots, m \quad (1) \\ & \sum_{k=1}^m Q_k x_{jk} + z_j \geq D_j \quad j = 1, \dots, n \quad (2) \\ & \sum_k x_{jk} \leq \lceil D_j / Q_{\min} \rceil \quad j = 1, \dots, n \quad (3) \\ & x_{jk}, z_j \in Z^+ \quad j = 1, \dots, n; k = 1, \dots, m \quad (4) \end{aligned}$$

where x is the smallest integer not less than x .

The objective function represents the total drying and buying costs. Constraint set (1) defines the lateness of each kiln charge where the maximum lateness is set to be less than or equal to zero (i.e., no jobs will be tardy). Constraint set (2) specifies that the production of each job (i.e., species/thickness) plus the amount of wood bought from an outside supplier is greater than or equal to the demand. Constraint set (3) defines an upper bound on kiln charges for a specific job. Constraint set (3) is not necessary theoretically, but does aid the Branch and Bound process in Integer Programming packages by providing additional cutting planes.

In terms of the problem studied here, the inclusion of drying and buying costs only adds complexity, as does non-zero availability times for

each kiln. The problem is easily reduced to the problem of scheduling n independent jobs on m non-identical parallel machines problem by assuming that all costs are zero, all kilns are available at time zero, and lateness (tardiness) of each job is non-negative. Since the latter problem has been shown to be NP-complete, including costs and kiln availability is also NP-complete.

Heuristic: The computational complexity of this problem makes it difficult to obtain optimal solutions for realistically sized problems within an appropriate time interval using commercial IP packages. The development of a heuristic approach to the problem is motivated. A three-stage heuristic based on a simple 3-opt-like procedure for exchanging jobs among kilns (Figure 1) is developed. Recall that the problem is reduced to the allocation of jobs among kilns since due-date ordering is a property of the optimal solution.

In the first stage, the algorithm finds a quick and dirty solution by randomly “charging” the kilns. It first chooses a kiln at random and then tries to place a portion (job) of a randomly chosen demand in this kiln while maintaining non-positive maximum lateness. Note that the quantity of a job cannot be more than the kiln capacity. If lateness will be positive, then another kiln is tried. The algorithm moves to the next stage if it cannot place any more jobs in any kiln in $2n$ or $3n$ (random choice) tries where n is the number of demands. Any demand not completely assigned to a kiln is assumed to be out-sourced.

In the second stage, the algorithm improves the first stage solution by using four switching routines which deal with the remaining demand for each job (i.e., the part of the demands not yet scheduled in the kilns). Within each second-stage switching routine the algorithm takes a job with some positive remaining demand and checks conditions for a possible successful switch (i.e., reduction in the cost function). When a switching rule is applied, it is tested for all possible combinations of jobs until a successful switch can be made. In the first routine, (2-1 Exchange), one kiln charge of a job with remaining positive demand is switched with two (already scheduled) charges of another job. The second routine, (1-2 Hanging Exchange), puts a job, j_1 , with a positive remaining demand in the place of an (already scheduled) job, j_2 , in some kiln k_j , and then

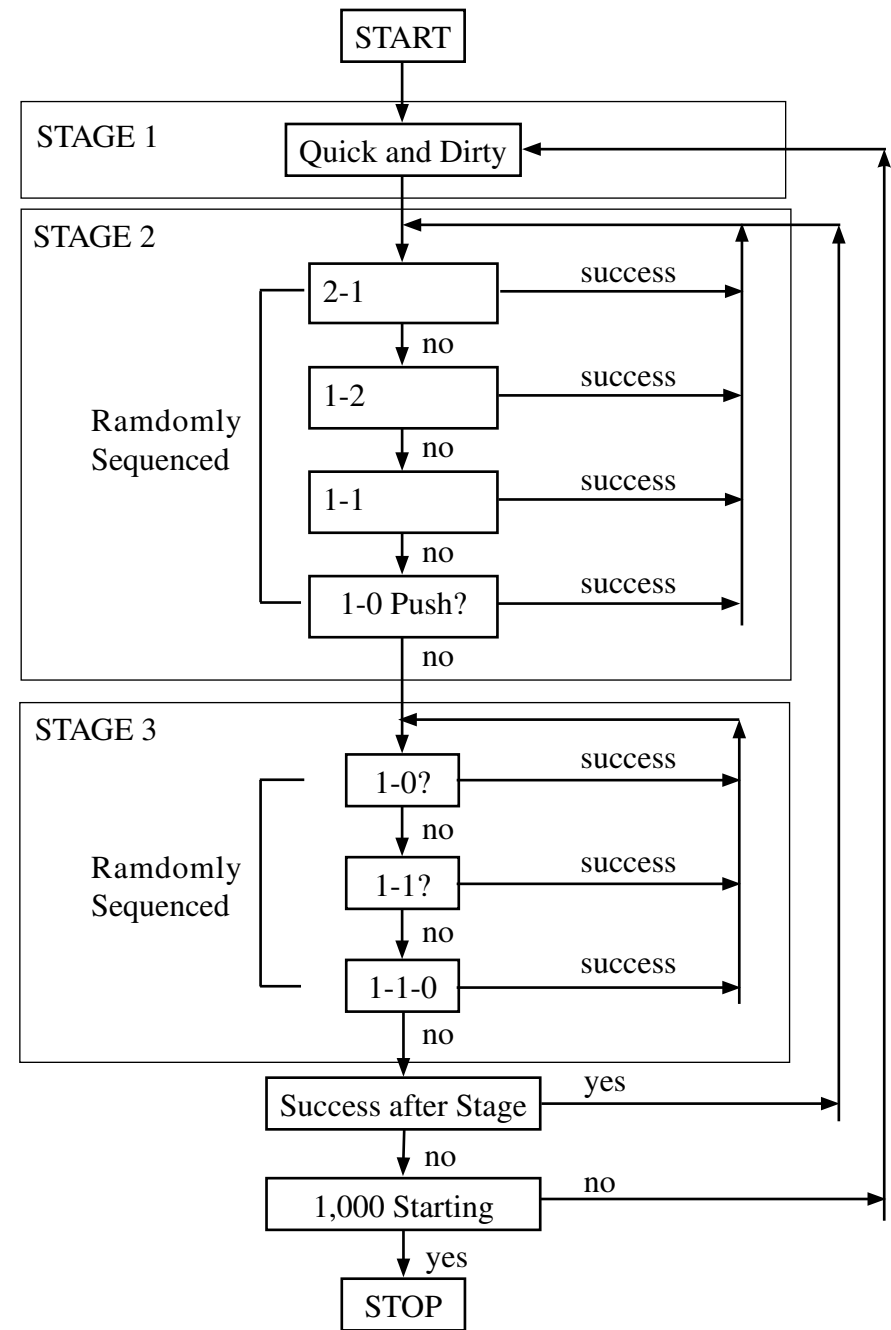


Fig. 1 Algorithm flow chart of the algorithm.

allocates that job, j_2 , to yet another kiln, k_2 . In the third routine, (1-1 Exchange), a job with remaining positive demand is switched with another (already scheduled) job. In the last switching routine, (1-0 Push), a job with a positive remaining demand is allocated to a kiln. Again, a switch is performed only if it results in a reduction in total cost and the demand and lateness requirements are still satisfied. Note that the routine's sequence of application is chosen randomly for each iteration (or starting point).

In the third stage, the algorithm “shakes” the solution. It tries to improve the solution by changing the allocation of the jobs among the kilns (i.e., the algorithm performs a “more local” search). In this stage, the out-sourced demands are not included (i.e., switching routines work only within the kilns). When a switching rule is applied, it is tested for all possible combinations of jobs until a successful switch can be made. The first routine (1-0) moves one charge from its current kiln to another kiln. The second routine (1-1) exchanges a charge from its current kiln with one from another kiln. The third routine (1-1-0) moves a charge of job j_1 from a kiln k_1 to another kiln, k_2 , and then removes a charge of another job from kiln k_2 and adds it to the list of out-sourced demand. Note that in case of the improvement of the objective value in the third stage the algorithm goes back to the second stage. Again, the switching routines' sequence of application is chosen randomly. The user specifies the number of permutations (i.e., the number of starting points).

Experimentation: The algorithm's performance was compared with the LINDO integer-programming package (IP). Twenty-five problems (up to 14 jobs and 5 machines) were randomly generated in the first stage of experiments. Each problem was solved by both the heuristic and by the IP on a DEC Station 3100. The time to compute 1,000 monkey trials of the heuristic took from 0.1 to 1.7 minutes. As it turns out, the heuristic found the optimal solution for each test problem. The time comparison of the heuristic to perform 1,000 permutations versus the IP to find the optimal solution is shown in Figure 2.

For larger problems, the computation times for the IP rapidly increased to days (rather than hours or minutes) without fully fathoming the branch and bound tree. The heuristic handles problems of a more realistic size relatively easily. Computational results for three large

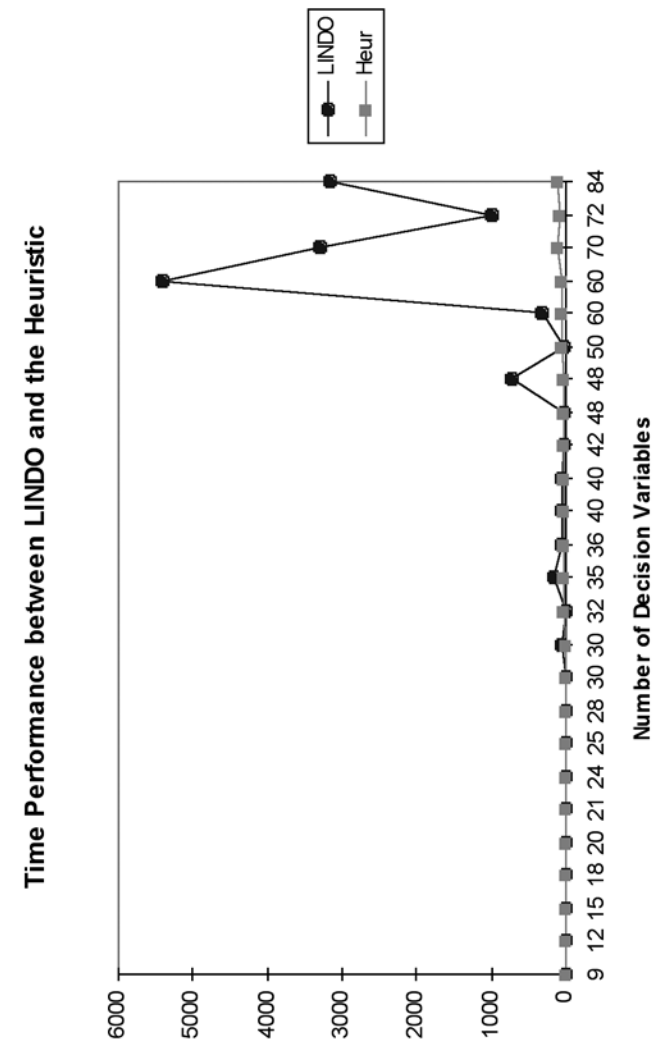


Fig. 2 Time performance between the IP and the heuristic.

problems are shown in Table 1. Note that ten replications of the heuristic were performed, and the mean and standard deviation of the objective is reported. This shows the increasing time/performance gap between the heuristic procedure and commercial optimization software in solving larger and more complicated versions of the problem.

Table 1: Heuristic versus IP for Large Problems

Technique	Problem Instance →	50 jobs, 5 kilns	75 jobs, 10 kilns	90 jobs, 12 kilns
IP	Lower bound	38,662	81,384	177,200
	Best solution found	74,777	196,429	358,980
	Time (DEC 3100)	72 hr.	72 hr.	72 hr.
Heuristic	Best solution found	40,039	144,459	223,446
	Average solution	40,144	148,489	231,601
	(std. dev.)	(145)	(4,508)	(5,700)
	# of starting points run	300	300	200
	Time (200 mhz PC)	10 min.	1 hr. 40 min.	2 hr. 20 min.

The above problems were randomly generated. This randomness might explain the effectiveness of the heuristic. The next experiment helps to determine the performance of the heuristic over a wide range of characteristics for one problem (ten jobs and four kilns). The size of problem is based on the computational time required to solve it optimally, which ranged from 25 seconds to 3 minutes. From previous work at classifying job shop problems, it is clear that the most important measure of difficulty for a problem is the range over which the due dates are spread. Ten instances of a ten-job, four-kiln problem were randomly generated

for nine different due-date ranges. Due dates were generated using

$$d_j = C + R * U(0,1),$$

where R represents the range, C is a constant, and $U(0,1)$ is a uniform random number between 0 and 1.

The heuristic shows excellent performance. The heuristic solved 97 out of 100 problems optimally. The average error was less than 0.07%, and in the worst case the heuristic was less than 1.4% off the value of the optimal solution.

Conclusion: An integer linear programming model for the cost analysis of a dry kiln scheduling problem has been developed. Drying and buying costs, kiln availability, and kiln capacity have been incorporated into the model. Owing to the complexity of the problem, the development of a randomized heuristic approach was motivated. A three-phase local improvement heuristic based on procedure for exchanging jobs among kilns was developed. Optimal solutions to a wide range of problems were compared with solutions obtained by the heuristic. Comparison indicates that the heuristic offers not only a tremendous advantage in speed but also provides high quality solutions. The monkey got the optimal solution in over 97% of the test cases that were solved to optimality by the IP.

Example 4: Satisfying Due Dates in Large Multi-Factory Supply Chains
with Kristin Thoney, Russell King, Mehmet Taner, Amy Wilson, Denis Cormier, Sasha Weintraub, Andy Zozom, Col. Tim Trainor, LTC Barbra Melendez, Scott Schultz

There were many separate pieces in this example. The names of all who worked on the pieces are included above.

A job shop scheduling procedure for conventional processors was expanded to include batch processors. Batch processors include batch heat treat operations, some painting and plating, and other batch chemical operations. Batch processing also includes both internal and external bulk

transportation operations. Including batch-processing operations in the scheduling model enables the detailed scheduling of entire multi-factory manufacturing/supply chains. We focused on the integrated scheduling of external transportation and conventional (job shop) factory systems. The procedure is sufficiently efficient as to allow detailed scheduling of large-scale manufacturing/logistic supply chains in near-real time.

We assume that a transportation vehicle does not leave until it is full or there are no more jobs left up stream in the system. This is a simplifying assumption but is consistent with what happens in industry when the batch processors are vehicles since full loads minimize transportation costs. This is not always the case, particularly when the batch processors are not vehicles.

Sequencing for the N-job/M-machine/Maximum lateness Problem:

The approach consists of repeatedly simulating the system to be scheduled while simultaneously updating job sequences based on the results of the previous simulation. During the first iteration, jobs are sequenced on machines in order of increasing slack. Let d_i be the due date of job i and p_{ij} be the processing time of job i on machine j . The slack of job i on machine m is computed as

$$\text{slack}_{im} = d_i - \sum_{j \in m+} p_{ij}$$

where $m+$ is the set of all operations on job i 's route subsequent to the one performed on machine m . Slack is a measure of the amount of time a job can queue and still meet its deadline. In general however, dispatching in order of increasing slack may not provide good results. This can be attributed to the fact that slack does not take queuing time into account.

In subsequent iterations, the queuing time of the previous iteration is used to modify slack, and jobs are sequenced in order of revised slack. Let q_{ij} be the queuing time of job i at machine j . Revised slack is computed as

$$\text{slack}'_{im} = d_i - \sum_{j \in m+} p_{ij} - \sum_{j \in m+^*} q_{ij}$$

where $m+^*$ is the set of all operations on job i 's route subsequent to the one performed on machine m except the one immediately following m . The slack calculation represents an intermediate due date. It is the time at which the job needs to begin processing at the next machine to meet its due date. This definition was found to be most effective in the procedure.

The procedure is run for a fixed number of iterations and the best solution is saved. The procedure tends to converge monotonically over the first 10 iterations or so for all size problems. In other words, the queuing time estimates become more accurate after the first iterations, producing better and better solutions. After that, solutions tend to 'bounce' with no particular pattern involved. However, if you run the process longer, you typically get better solutions. In the present case, 200 iterations were used. While we have been able to achieve L_{\max} equal to the lower bound on several problems using real data, we have not been able to identify the general conditions required for this to happen.

A Lower Bound for N/M/ L_{\max} : Since the N/M/ L_{\max} problem is NP-hard, the optimal solution may not be known, especially for large-scale problems. Fortunately, an effective lower bound (*LB*) for the problem can easily be computed. Comparing the solution of a scheduling procedure to the *LB* gives the maximum amount by which the solution exceeds the optimal. The lower bound procedure is not discussed here.

Queuing Schemes for Batch Processors: The manner in which queuing time is used in the scheduling process for conventional processors is inappropriate for batch operations. Queuing time for a job on a conventional machine is computed as the time the job starts processing on the machine minus the time at which the job arrived at that machine. In the procedure outlined above, queuing time incurred increases the priority of upstream operations in the next iteration of the scheduling procedure. However, prioritizing the jobs by increasing revised slack does not account for the way they interact in batch processing. To see this, consider the following example for a batch processor. Suppose there is a batch processor of capacity two. The first two jobs that arrive at the batch processor, jobs 1 and 2, arrive at times 10 and 15, respectively. The batch processor begins processing the jobs at time 15. For a conventional

processor, job 1 would incur 5 units of queuing time and job 2 would incur 0 units. Job 1's priority for upstream operations would increase while job 2's would not. However, for a batch processor, having job 1 arrive earlier does not help if job 2 still arrives at time 15. Instead, it is desirable for job 2's priority to increase so that it may arrive earlier and the machine can start processing the batch sooner. Thus, the use of queuing time, as applied to conventional processors in the system (i.e., for computing revised slack), appears inappropriate when applied to batch processors.

Based on this insight, two schemes to compute equivalent queuing time for batch operations in parallel have been devised. The following notation is required.

- Q_i = queuing time for job i at the batch processor
- JA_i = time job i arrives at batch processor's queue
- BF_i = time the machine on which job i is processed finished its previous batch
- BA_i = time the next batch begins processing after the arrival of job i
- BB_i = time the previous batch began processing before the arrival of job i

It is important to note that BB_i and BF_i may or may not refer to the time at which the same batch processor begins and ends processing. BB_i is not necessarily related to the batch processor on which job i will be processed as BF_i is. BF_i can be thought of as the time at which the batch processor could begin processing job i 's batch if all the jobs had already arrived in the batch processor's queue. Note that BA_i does not necessarily refer to the actual starting time of the machine on which job i 's batch is processed.

Figure 1 illustrates these terms in a two-vehicle example. The relationship between JA_i and BA_i and BB_i holds regardless if job i is transported by vehicle 1 or 2. If job i is transported by vehicle 1, then $BF_i = x$. If job i is transported by vehicle 2, then $BF_i = BB_i$.

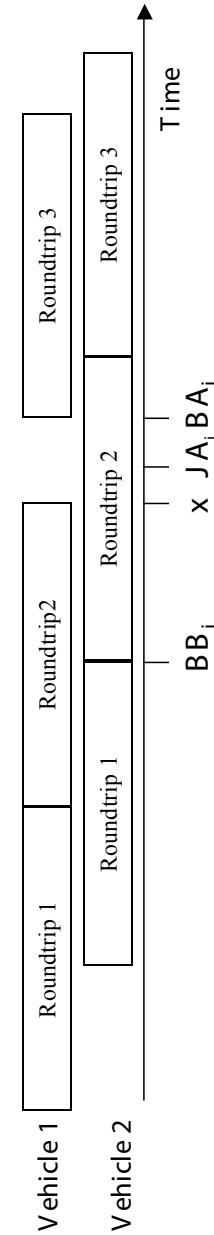


Fig. 1 Queuing scheme definition example.

The queuing schemes are as follows.

1. $Q_i = \max\{JA_i - BF_i, 0\}$ If the job arrives after the machine finished its previous batch, then the job should be there earlier. The priority of the job for upstream operations is increased by allocating a positive queuing time, equal to the difference between the job arrival time and the time at which the machine finished its previous batch. If the job arrives before or at the same time as the machine finished its previous batch, then the job arrival time is okay and thus is given zero queuing time.

2. $Q_i = \begin{cases} JA_i - BB_i & \text{if } JA_i \leq (BA_i + BB_i)/2 \\ 0 & \text{otherwise} \end{cases}$ If the job arrives prior to the midpoint of the time between the previous batch and the next batch began processing, it may be advantageous for the job to be processed in the previous batch instead of one of the jobs that was processed there. The priority of the job for upstream operations is increased by allocating a positive queuing time, equal to the difference of the job's arrival time and the time at which the previous batch began processing. If the job arrives after the midpoint, it is given zero queuing time.

Queuing scheme 1 is fundamentally different than queuing scheme

2. Scheme 1 tries to force the machine to begin processing as soon as possible. Scheme 2 tries to place a job in the right batch. Both objectives have merit, but scheme 2 tends to dominate scheme 1. The two schemes perform similarly when due date ranges are high, but scheme 2 tends to be better when due date ranges are low. Scheme 2 is used in all experimentation.

Experiments with Two Factories in Series

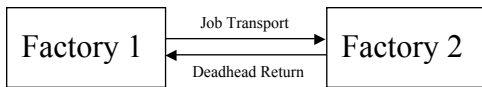


Fig. 2

Problem Generation: In this section, problems with 50 machines in each factory are considered. Trucks have a capacity of 10 jobs. One way travel time for trucks is 250. Five hundred jobs are each assigned a number of operations, w (randomly generated), that is Uniformly distributed [1-11] and a corresponding (Uniformly distributed) due date. If $w > 6$, then the job is processed on $w-6$ machines in factory 1, transported by truck to factory 2, and processed on 5 machines in factory 2. If $w = 6$, then the job is transported by truck to factory 2 where it is processed on 5 machines. If $w < 6$, then the job is processed on w machines in factory 2. This implies that, for each problem generated, $500/11$ 45.5 jobs are expected to have each of the possible number of operations (1-11). In addition, $500(5/11)$ 227.3 jobs are expected to start in each of the two factories, and $500(6/11)$ 272.7 jobs (approximately 28 truckloads) are expected to be transported by truck from factory 1 to factory 2.

Performance Observations: In Figure 3, the difference between the attained L_{max} and the lower bound (LB) is displayed as a function of the due date range and number of trucks. It can be seen that performance is close to the lower bound (i.e., $L_{max}-LB$ peaks at 250 and is less than 100 over almost the entire due-date range) for almost all R where there are 9 or more trucks. Performance is close to the lower bound in the low due-date ranges for any case. Performance for 1-5 trucks is similar in the high due-date ranges. Performance in the high due-date ranges progressively improves for 6-8 trucks. For greater than 9 trucks, transportation is not a constraint and overall performance of the scheduling procedure is virtually identical to performance of a normal single job shop.

Estimating the Truck Bottleneck: The point at which trucks are no longer a bottleneck occurs when the average inter-factory transportation rate of the trucks is greater than or equal to the output rate of factory 1. To estimate this point, a number of intermediate calculations must be made. The earliest expected time that factory 1 could be empty can be expressed by dividing the total expected amount of time that jobs need to be processed in factory 1 equally among the machines in factory 1. That is

$$E_1 = \frac{N \cdot \overline{ops} \cdot \overline{P}}{M} = \frac{[5/11 \cdot (500)](3)(100.5)}{50} \cong 1370$$

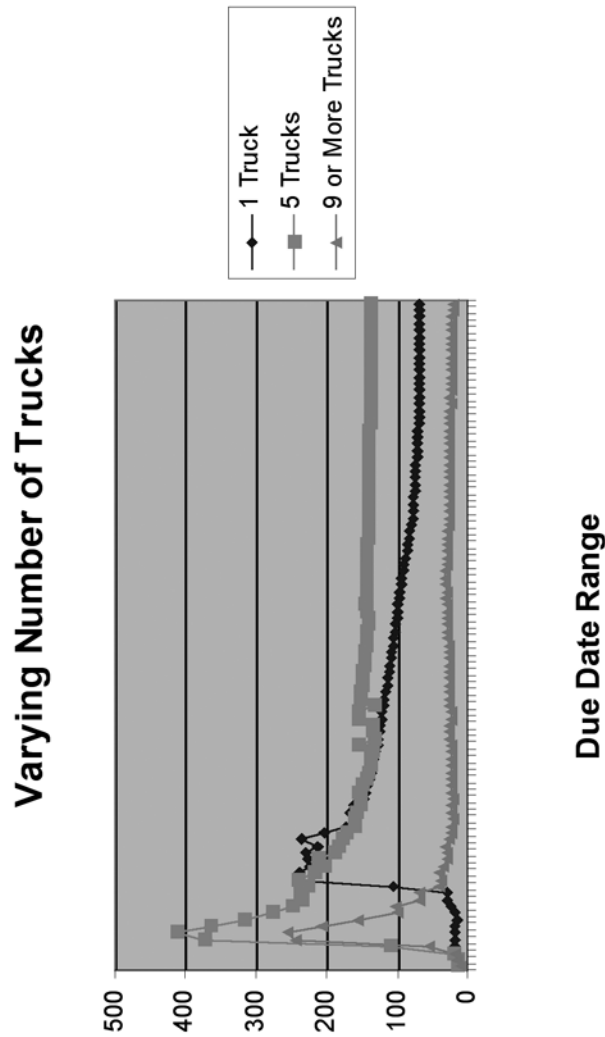


Fig. 3 Two factories in series.

where N is the expected number of jobs in factory 1, \overline{ops} is the expected number of operations for a job in factory 1, \bar{p} is the expected job processing time, and M is the number of machines in factory 1. The fastest average rate that jobs could finish in factory 1 is one every

$$\frac{E_1}{N} \cong \frac{1370}{[5/11 \cdot (500)]} \cong 6.0$$

units of time. Therefore, on the average a new truckload (capacity = 10 jobs) is ready for loading at factory 1 every $6.0(10) = 60$ time units. To handle this rate, when the round trip travel time for each truck is 500 time units, approximately $[500/60] = 9$ trucks are needed. This is consistent with the observed data in that adding more trucks beyond 9 does not affect the solutions (see Figure 3). The estimate tends to be less accurate on non-homogeneous (real) data. However, we have found it to be a useful computation in characterizing both generated and observed data sets. Computing the point at which trucks are no longer a bottleneck in more complex manufacturing systems, is a direct extension of “two factories in series.”

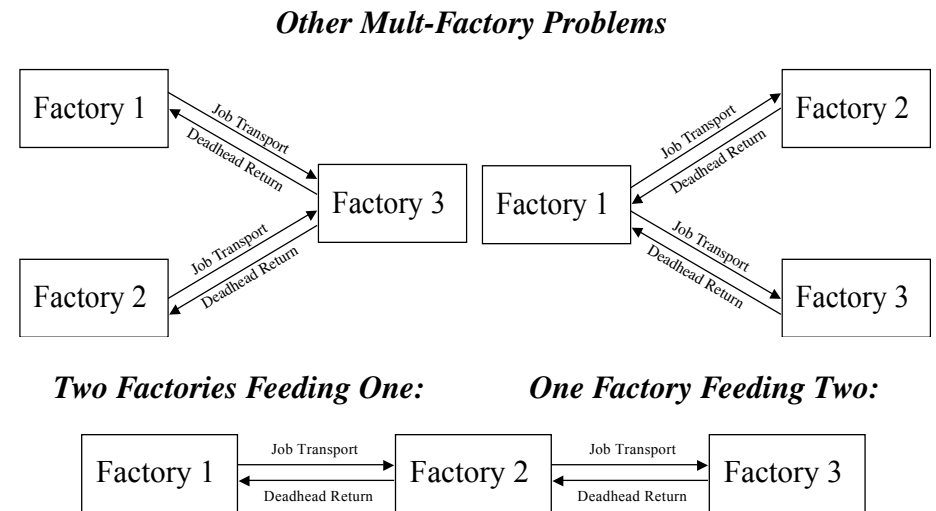


Fig. 4 Three factories in a series.

Observations: The performance of “2 factories feeding 1,” “1 factory feeding 2,” and “3 factories in series,” while not identical to the case of “2 factories in series,” is virtually the same. Thus, graphical results are not presented here. The point is that the methodology presented is applicable to complex multi-factory scenarios at no loss.

Computational Effort: Figure 3 represents 10,000 problem solutions. Each problem requires 2-4 seconds on an 850 MHz PC to solve. The methodology is linear in the number of operations in that doubling the total number of job operations generally doubles the computation time. Five thousand job problems generated along the same scenario as here run in approximately 30 seconds.

Conclusions: A methodology for the detailed scheduling of large-scale manufacturing/logistics supply chains has been developed. The scheduling system performs well relative to the lower bound in scenarios where transportation is not a bottleneck. In the industrial situations we have observed, transportation is seldom a bottleneck. A transportation shortage is generally, at worst, a transient situation.

Most importantly, computational effort is linear in the number of job operations, and large-scale problems of industrial size can be solved quickly. The system provides the planner/scheduler with a “look-ahead” tool that can identify resource issues both in manufacturing and in transportation.

A Final Comment: We have seen four problems that because of their difficult structure are difficult to solve directly. The monkey with a typewriter, however, gives us a way to find extremely good solutions to these problems. Out of chaos comes order.

The R. J. Reynolds Tobacco Company Award Distinguished Lecture Series

1. *The Role of Solid-State Research in Electrical Engineering* by John Reid Hauser, Professor of Electrical Engineering, North Carolina State University, April 27, 1982.
2. *Does Engineering Education Have Anything To Do With Either One?* by Richard Mark Felder, Professor of Chemical Engineering, North Carolina State University, October 12, 1982.
3. *Materials and Devices for Optical Fiber Communications* by Michael Anthony Littlejohn, Professor of Communications and Signal Processing, North Carolina State University, November 10, 1983.
4. *Two Theories of Communication* by John Benjamin O’Neal, Jr., Distinguished Professor of Communications and Signal Processing, North Carolina State University, October 31, 1984.
5. *On Zero and Risk* by Harold B. Hopfenberg, Camille Dreyfus Professor and Head of the Department of Chemical Engineering, North Carolina State University, October 29, 1985.
6. *An Overview and a Point of View on Heat Transfer* by M. Necati Ozisik, Professor of Mechanical and Aerospace Engineering, North Carolina State University, October 29, 1986.
7. *On Quality, Automation, Cultural Relativism, and Things Like That* by Salah E. Elmaghraby, University Professor, Operations Research and Industrial Engineering, North Carolina State University, November 12, 1987.
8. *Humans to Mars* by Fred Roark DeJarnette, Professor of Mechanical and Aerospace Engineering and Director of Mars Mission Research Center and Hypersonic Aerodynamics Program, North Carolina State University, November 9, 1988.

9. *The Time Has Come* by Carl Frank Zorowski, Professor of Mechanical and Aerospace Engineering and Director of Integrated Manufacturing Systems Engineering Institute, North Carolina State University, November 15, 1989 (printed lecture unavailable).
10. *Science is . . . Sensual* by Ruben G. Carbonell, Hoechst Celanese Professor of Chemical Engineering, North Carolina State University, November 7, 1990 (printed lecture unavailable).
11. *Education, Research, and Entropy* by Ronald O. Scattergood, Professor of Materials Science and Engineering, North Carolina State University, November 13, 1991 (printed lecture unavailable).
12. *Research Is Teaching* by Salah M. Bedair, Professor of Electrical and Computer Engineering, North Carolina State University, November 4, 1992.
13. *The Role of Monte Carlo Methods in Engineering Education* by H. A. Hassan, Professor of Mechanical and Aerospace Engineering, North Carolina State University, November 3, 1993.
14. *Radioisotope and Radiation Measurement Applications* by Robin Pierce Gardner, Graduate Alumni Distinguished Professor of Nuclear and Chemical Engineering and Director of the Center for Engineering Applications of Radioisotopes, North Carolina State University, October 28, 1998.
15. *Silicon Carbide, Diamond, and Gallium Nitride: Sources for New Electronic Materials, New Gemstones, and New Corporations* by Robert F. Davis, Kobe Steel Ltd. Distinguished University Professor of Materials Science and Engineering, North Carolina State University, October 27, 1999.
16. *Trends in Power Discrete Devices* by B. Jayant Baliga, Distinguished University Professor of Electrical and Computer Engineering, Director of the Power Semiconductor Research Center, North Carolina State University, November 1, 2000.
17. *On Habitual Domains, Fuzzy Sets, Variational Inequalities, and Optimization* by Shu-Cherng Fang, Walter Clark Professor of Industrial Engineering, Director of Graduate Programs in Industrial Engineering, North Carolina State University, November 14, 2001.
18. *Going to Extremes: Observations from the Biology/Engineering Interface* by Robert M. Kelly, Alcoa Professor of Chemical Engineering and Director of the North Carolina State University Biotechnology Program, January 30, 2003.
19. *Lifelong Learning and Teaching in a Changing Profession* by Carl C. Koch, Professor of Materials Science and Engineering, November 19, 2003.
20. *Thinking Like a Molecule: Computer Simulations of Protein Aggregation* by Carol K. Hall, Alcoa Professor of Chemical Engineering, October 13, 2004.
21. *Monkey with a Typewriter: Solving Problems in Industrial Engineering* by Thom J. Hodgson, Distinguished University Professor of Industrial Engineering, James T. Ryan Professor of Industrial Engineering and Furniture Manufacturing, and Director of the Integrated Manufacturing Systems Engineering Institute, October 20, 2005.

The Award

The R.J. Reynolds Tobacco Company Award for Excellence in Teaching, Research, and Extension was established at the College of Engineering to honor a member of the engineering faculty who has demonstrated superiority in several areas of activity that relate to the University's three-fold mission of teaching, research, and extension.

The annual award is supported by R.J. Reynolds Co. through the NC State Engineering Foundation, Inc. to bring recognition to scientific and educational achievements in the field of engineering.

All engineering faculty members are eligible for the award. Nominations may come from any individual or group. A committee of engineering faculty, representing either academic, research, or extension areas, and representatives from other University schools reviews the nominations and recommends a candidate to the Dean of Engineering.

The following criteria are used in the selection of the candidate:

- Contributions to the education of students through excellent teaching, course and program improvements, advising, student project and thesis direction, and participation in other activities that enhance the development of students.
- Scholarly activities, including research, design, writing, and speaking that are recognized locally, nationally, and internationally.
- Extension and public service activities, including assistance and advice to industry, business, government, and other educational institutions, and contributions to professional societies.

The award recipient is asked to prepare a lecture on a topic related to the individual's engineering activities, to be delivered during a special ceremony honoring the recipient with a monetary gift and certificate.

Each lecture is published under a series bearing the title, "The R.J. Reynolds Tobacco Company Award Distinguished Lecture Series." The publications are available upon request to the Office of the Dean.