Decision Support Systems 9 (1993) 245–257
North-Holland

A mixed-initiative scheduling workbench

Integrating AI, OR and HCI *

Wen-Ling Hsu, Michael J. Prietula and Gerald L. Thompson
Carnegie Mellon University, Pittsburgh PA, USA

Peng Si Ow
IBM, Austin TX, USA

In this paper we describe a decision support system for scheduling called *MacMerl*. This system weaves together numeric and symbolic computing techniques to form a 'scheduler's workbench'. MacMerl has two major components. The first is a Scheduling Kernel which includes a Generative Scheduler, a Constraint Checker, and a Reactive Scheduler. The second is a Manual Scheduler which permits the human to create or modify schedules and includes a Critiquer as well as access to routines in the Scheduling Kernel. Taken together, these components support an approach to problem solving we call *mixed-initiative scheduling* in which the human and the machine interact in a coherent and cooperative manner to solve complex production scheduling problems.

Keywords: Decision support systems, Scheduling, Human–computer interaction.

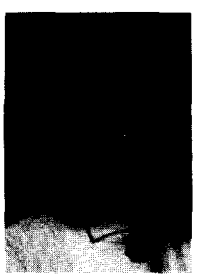
> It might appear that there are natural algorithms, or step-by-step procedures, for constructing highly efficient schedules. That, however, is not the case. Apparently logical ways of constructing schedules cannot be counted on to perform equally well in different situations... Some of the commonest and most intuitive scheduling procedures can give rise to unexpected and even seemingly paradoxical results. [3, p. 24].

* This work was funded through an external corporate grant to the Center for the Management of Technology, Graduate School of Industrial Administration, Carnegie Mellon University. We would like to express our appreciation to the Center's director, Paul Goodman, for his guidance and advice on this project. We would also like to thank the corporate participants for their time and effort. Cameron Long and Sandi Kwee were the primary programmers for the project.

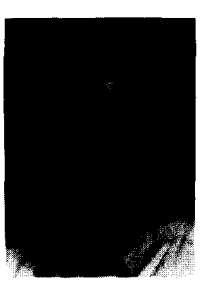
Correspondence to: W.-L. Hsu, Siemens Corporate Research Inc., Princeton, NJ 08540, USA.

1. Introduction

Scheduling is one of the most common and important activities in production. A poorly scheduled production environment can cause disruptions leading to serious erosion of a firm's ability to compete. Scheduling also encompasses some of the most difficult problems to solve. Every manufacturing plant which produces a variety of products in a factory with multiple ma-



Wen-Ling Hsu is an Assistant Professor of Management Information Systems in the School of Urban and Public Affairs at the Carnegie Mellon University. She received the B.S. degree in computer science from the National Chiao-Tung University, Taiwan and the Ph.D. degree in Management Information Systems from Krannert Graduate School of Management, Purdue University in 1986. She is currently on leave with the Decision Support Systems group in Siemens Corporate Research Inc., Princeton, New Jersey. Her areas of interest include decision support systems, applications of artificial intelligence in manufacturing and machine learning. Her current research involves integrating AI, OR and machine learning techniques for decision support in manufacturing environment. Dr. Hsu is a member of IEEE Computer Society, ACM and American Association for Artificial Intelligence.



Michael J. Prietula is an Assistant Professor of Information Systems in the Graduate School of Industrial Administration (GSIA) at Carnegie Mellon University. He received his Ph.D. in Management Information Systems from the University of Minnesota. He was a member of the Man-Machine Sciences Group at Honeywell's Systems and Research Center where he conduced research on human-computer interaction and artificial intelligence in the aerospace and defense arena. He has taught in the Amos Tuck School of Business Administration and the Program and Computer and Information Science at Dartmouth College. He continues his research in expertise and artificial intelligence in the Center for the Management of Technology at GSIA with colleagues in business, psychology and computer science.

chines and employing workers possessing different skills, faces such complicated scheduling problems. Many problems in job shop scheduling, for example, are NP-hard so that it is appropriate to use (perhaps) efficient, but approximate, techniques [6]. In practice, this means that scheduling behavior differs significantly from scheduling theory [5,10,14].

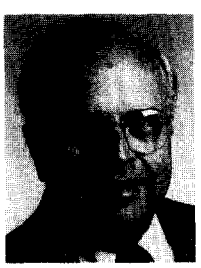
Two primary sources of scheduling difficulty in manufacturing environments are (1) the nature of the constraints that must be addressed and (2) the unpredictability of the environment. The constraints of a scheduling problem are partly technological, partly managerial, and partly directed to worker satisfaction and/or safety. Some constraints are "hard" and must be satisfied in order to obtain feasible solutions whereas other constraints are "softer" and can be violated if necessary. A proposed schedule must satisfy all of the hard constraints and as many of the soft ones as possible. In the case described in this paper, soft constraints, referred to as preferences, include: minimization of setup costs, minimisation of stockouts, minimization of inventory holding costs, interweaving long and short jobs on a machine, and minimisation of worker accidents.

Generated schedules are based on a series of assumptions corresponding to events in the real world: the timely delivery of raw materials, the smooth functioning of a set of machines, and a stable set of preference axioms identifying the relative importance of awaiting jobs. In many scheduling situations, however, unanticipated events can quickly occur which rapidly invalidate a schedule. If a scheduling system (human or machine) can react appropriately, adjustments can be determined to minimize disruption; otherwise, the scheduling effort itself risks losing both credibility and utility as schedules diverge from reality. Given these difficulties, the first (of four) observations concerning these types of scheduling problems can be offered:

Observation 1. Due to the complex and interacting nature of constraints, preferences, and unpredictability of the scheduling environment, practical scheduling tasks in many firms are too complex to be solved by explicit numerical methods alone.

Despite the problems, scheduling does get done. Most plant scheduling is accomplished by human schedulers who have worked in the plant environment for many years. These schedulers know, and have internalized, the relevant constraints and preferences from first hand experience. To produce a schedule, they generally rely on simple analytic techniques used in conjunction with the knowledge they have gained by spending 15 to 30 years at the task. Oftentimes, two or more schedulers are assigned to the scheduling effort (especially if multiple shifts are involved) to provide backup for sick leave and vacation. Because of the extreme importance of the scheduling problem, the loss of a scheduler (e.g., through retirement or job change) is an event that must be prepared for by the laborious, apprenticeship training of new personnel to fill the job. Therefore,

Observation 2. Scheduling is often based on the application of a scheduler's (or schedulers') extensive knowledge accumulated from years of experience on the task and, possibly, augmented with simple numerical methods.



Gerald L. Thompson is the IBM Professor of Systems ad Operations Research at the Graduate School of Industrial Administration of Carnegie Mellon University where he has been on the faculty since 1959. He is also the E.D. Walker Centennial Fellow at the IC^2 Institute at the University of Texas in Austin. He received a B.S. degree in Electrical Engineering from Iowa State University, an S.M. degree in Mathematics from MIT, and a Ph.D. in Mathematics from the University of Michigan. His research is in large scale linear ad quadratic programming, combinatorial optimization using parallel and distributed groups of computers, optimal control theory with management science applications, theory and applications of scheduling techniques, and computational economics, which is economic modelling and planning by using mathematical programming software.

Peng Si Ow is a senior engineer at the International Business Machines Corporation, in the Entry Systems Division. She received her B.Sc. (Joint Honours) in Computer Science and Accounting from the University of Manchester, and her M.Sc. in Systems Sciences and Ph.D. in Industrial Administration from the Graduate School of Industrial Administration at Carnegie Mellon University. She was a member of the faculty at the Graduate School of Industrial Administration. Her work has been primarily in the area of information systems and operations management.

In tasks like this, experience is critical for the development of useful knowledge but can take many years of practice. In effect, the scheduler becomes a "special purpose machine" and has adapted to the task in ways that overcome cognitive limitations and permit effective scheduling performance that is, the scheduler has become an expert [15]. The power of the expert's reasoning methods is based on the development of knowledge structures and heuristics that permit effective and efficient access to the appropriate knowledge in response to the demands of the scheduling environment. The nature of such adaptations (i.e., the kinds of knowledge structures, strategies, heuristics, etc.) reflects the critical elements and influences of the scheduling task. Therefore, by studying how the expert adapts to a complex, problem solving task, we can gain insight into ways to design a system to implement or augment these adaptations. Consequently,

Observation 3. The scheduling knowledge noted in Observation 2 is an asset of a firm which provides insights into the scheduling task; therefore, the design of a human-computer scheduling system should include the most valuable aspects of that knowledge.

With the advent of powerful personal computers and artificial intelligence techniques (AI), it is possible to combine symbolic computing methods from AI with the numerical methods of operations research (OR) scheduling techniques in a system that exploits the structure of a given problem by emulating the human scheduler with symbolic methods when appropriate, yet invoking more powerful numerical techniques when possible [8]. Efforts to represent such knowledge are indeed yielding insights into possible architectures for AI scheduling systems and assistants [4,10,16]. Research into this type of problem is embryonic and the results are not yet unequivocal. We have no taxonomy to enable us to map problem characteristics onto AI solution techniques in an unambiguous manner – new techniques are constantly being provided and we are still attempting to understand the power and limits of the existing ones. However, the success and insights we have gained permit us to make a final observation:

Observation 4. To the extent that the scheduling environment is sufficiently stable for human expertise to develop and be exploited for the construction of schedules, a system can be configured which can function effectively as a "knowledgeable colleague" to help in the solution of complex scheduling problems.

MacMerl is such a scheduling system for producing and evaluating schedules in an automotive replacement glass factory in Western Pennsylvania. The scheduling problem embodies a number of interesting characteristics for HCI (human computer interaction), OR (operations research) and AI (artificial intelligence). First, because the human scheduler actively participates in the scheduling function, the system must serve as a colleague and be able to make comments and observations that are understandable to, and consistent with the knowledge possessed by the human scheduler. In order to do this it is necessary to study and to understand the expertise of the scheduler. Second, there is a primary tradeoff between structures and algorithms that support fast access and execution speed versus those that can easily embed symbolic elements reflecting more closely the human scheduler's knowledge and strategies. Finally, the system interface is the keystone for the system design. The "interface" not only refers to what is presented on a computer screen, but how the scheduler is able to interact with the displays and the requisite structures of the computer program underlying the displays and interaction. To the user, the interface is the system. Fundamental to this is the knowledge engineering effort that has yielded insights to the search and representation strategies as well as the structure of interface design itself.

2. The scheduling task environment

The scheduling task concerns the production of replacement windshields for automobiles (see fig. 1). The primary raw materials for producing windshields are glass and vinyl, where each windshield consists of two pieces of glass, one piece of vinyl, and an edge strip. The glass needs to be cut and bent, and the vinyl needs to be stretched and cut before the two can be put together. Further operations such as deairing, laminating, and attaching the ancillary parts (e.g., rear-view mirror

attachments) are applied before the finished products can be added to the inventory.

The most critical phase of production is the bending process, where two pieces of matched automotive glass are placed on a special rack-holder, called an iron, and moved through a large oven, called a lehr. As the glass is heated, it softens and bends in a certain shape determined by the interaction of gravity, counterweights on the iron, the particular heating pattern imposed on it, the speed of its movement through the lehr, and attributes of the glass itself (e.g., thickness, composition). It continues on the conveyor out of the oven and around the plant to cool before any further assembly can be done. In the plant, the task of scheduling jobs to the lehr for bending the raw glass is the bottleneck activity. This particular scheduling task includes deciding which type of windshield glass should be bent at what time and for how long (the length of the time determines the quantity of glass to be bent as well).

Sometimes, two different types of windshield glass can be bent in the same lehr setting (e.g., temperature, humidity, pressure, speed) and therefore can be scheduled to run through the lehr at the same time period without a new setup. However, no more than two different type of windshields can be scheduled in the same time. Setups in this process are generally expensive and require on the average of 45 minutes of down-time for the ovens due to the exchange of iron racks and manipulations of oven settings. On the other hand, if two different, but compatible windshields can be scheduled back-to-back, then the setup time is minimized as oven settings are retained and the exchange of iron-racks can be done "on the fly."

At the plant, scheduling of the lehrs has been accomplished by two employees who have over thirty years of combined experience. Throughout those years, they have developed and relied on effective heuristic methods and tactics to schedule lehr production runs. For the development of MacMerl, the most experienced of the two schedulers (who, in fact, now has primary responsibility for the generation of the schedules) was selected to serve as the referent expert.

The scheduling task itself involves the determination of when a part is to be run on which lehr (oven) and for what duration. The human scheduler constructs these schedules using a rolling horizon procedure: each week a new 5-week schedule is generated which makes use of the 5-week schedule constructed in the previous week, the production of the previous week, and an up-to-date windshield inventory report. Although the bending process is the primary concern, considerations are also given to other required operations (such as materials ordering and preparation) as well as the scheduling of labor. As noted earlier, there is a distinction between the sets of constraints which define admissible schedule options and preferences which reflect concerns that are "softer," but are essential in determining relevant precedence orderings over a set of admissible schedules.

For example, many of the heuristics the expert uses concerns the working conditions of the employees – large, heavy windshields tend to exhaust the workers and hence the interweaving of large and small jobs is desirable. This kind of preference knowledge is very difficult to factor into traditional scheduling problem solving, yet is fundamental to the formation, diagnosis, and re-

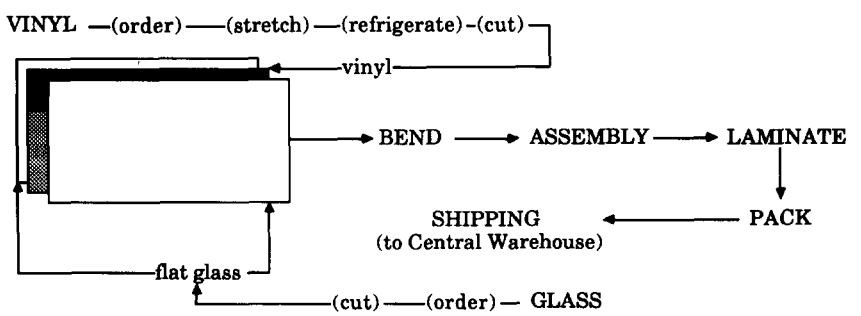


Fig. 1. Overall steps of the production process.

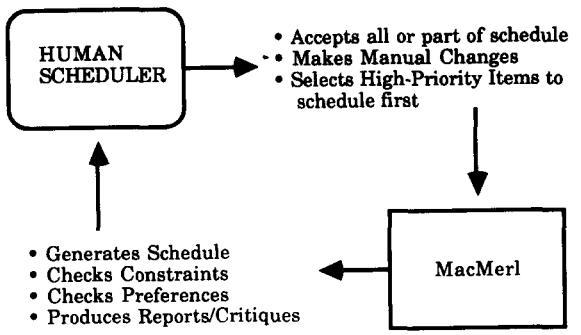


Fig. 2. Interactive process of schedule generation and review for MacMerl.

pair of schedules, and, consequently, essential to the adoption of scheduling techniques in practice [9].

The overall objectives of the scheduling process are the minimization of finished goods inventory, of stock outage, and of stock defects. These objectives were not stated explicitly, but were implicit in the methods used by the expert scheduler and were evidenced when the expert explained the method used to construct schedules. Our-approach was to develop a system that (1) replicated the most relevant components of the expert's strategies and (2) provided assistance to the expert (or any other human scheduler) in generating and evaluating schedules as well as reacting to the changes that occur in shop floor which disrupt existing schedules. This interactive process is illustrated in fig. 2.

3. Scheduling system description

The overall system architecture for MacMerl is given in fig. 3. MacMerl written in object-oriented "C" code which, when compiled, occupies approximately 300 Kbytes on an Apple Macintosh computer. The main components of MacMerl are the Scheduling Kernel and the Manual Scheduler. The Scheduling Kernel serves as the primary problem processing unit of the decision support system [1]. The development of the Scheduling Kernel was divided into three basic (not necessarily sequential) phases. The goal of the first phase was to duplicate the methods used by the human expert. The results of this phase was the initial version of the Generative Scheduler component, which included the Constraint Checker. The goal of the second phase was to incorporate intelligent search techniques to improve the performance of Generative Scheduler. In this phase, Reactive Scheduling components were also added to the Scheduling Kernel. These components assist in the generation of schedules and will eventually assist the human scheduler in reacting to unanticipated events which affect an existing schedule. We have designed the reactive scheduler so that it can serve as the foundation for the development of a totally automatic reactive scheduling environment. Current research is underway on testing how various algorithmic approaches in cooperation with human methods can yield a solution to this difficult scheduling task. Finally, a graphical user interface was provided which included a Manual Scheduler and served to integrate all the system routines in an icon-based, interactive environment. For the Manual Scheduler, the graphical user interface permits manual manipulation of the schedule and incorporates a Critiquer facility. Furthermore, all routines in the Scheduling Kernel are invocable from the Manual Scheduler.

3.1 Scheduling kernel

The development of the Generative Scheduler was based on the analysis of the expert. Initially, the goal was simply (or not so simply) to clone the methods of the expert and create a system that is capable of generating schedules similar to those generated by the expert. This was a long and arduous knowledge acquisition task which required many iterations; however, it was essential to instantiate the acquired knowledge in code in order to handle the same scheduling tasks that the expert was used to solving. To attempt to simplify the acquisition process or substitute another form of task other than the true scheduling task would have yielded inadequate results as it became apparent that the expert's knowledge was highly contextual to specific task cues. In fact, an experiment using this expert demonstrated how fragile such scheduling knowledge can be, where it occurs, and how it can be described in terms of cognitive components of deliberation [7].

The expert's general heuristics for both revising and creating a 5-week schedule were the following:

(1) Schedule parts with lowest inventory first to avoid stock-outs. (The expert also took into account the priorities of the parts – parts with a high stock-out cost have high priority.)
(2) Schedule as many as possible of the parts in a reservation-group close together in order to reduce set-up time.

The first heuristic is commonly used by "made to stock" manufacturers when no actual due-dates are assigned to parts. The second heuristic is also commonly used to reduce set-up costs. The concept of a "reservation group" refers to a set of jobs which can be run sequentially on the same machine without changing machine settings, which involves a setup cost. As a consequence, no machine down time is required. To accomplish this, a mapping is required from any given part to a set of other parts which are "reservation-compatible." However, in practice some parts can form a "better" reservation-group than others, but the quality of a reservation-group (and sometimes even membership) is quite difficult to specify analytically. Whether parts are reservation-compatible or not could be articulated by general rules; however, the determination that one part is "more compatible" than another had to be left to an explicit enumeration of cases. The concept of "compatibility" is embedded in a data structure which permits the rapid generation of those parts that are compatible with a particular part. Such compatibility data is important knowledge that

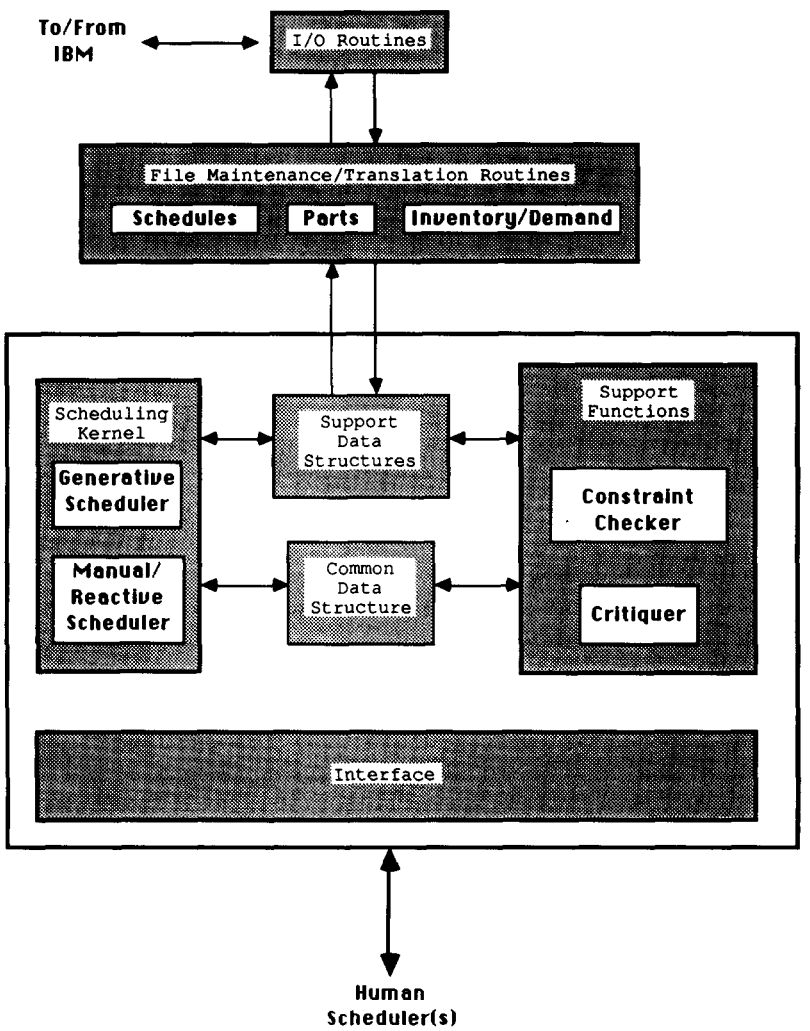


Fig. 3. System architecture for MacMerl.

can be explicitly stored and maintained by the company.

When constructing a schedule, the human scheduler also had other concerns, such as, "avoid scheduling difficult parts in the beginning and end of the week" (when absenteeism is highest) and "avoid scheduling large parts back to back" (to avoid the continued lifting of heavy windshields). It is not clear whether this breakage occurred because of physical strain or served as editorial comments on the quality of the Schedule. In any case, it was a rule that was rarely ignored. Observations revealed that the expert's overall approach to scheduling was mostly constraint-driven rather than trying to optimize an objective function. Therefore, it is possible to improve the performance of MacMerl by improving the search rules.

Generative Scheduler. The basic algorithm reflecting the expert's approach to forming schedules is incorporated in the Generative Scheduler. The Generative Scheduler decomposes the problem into three subproblems: selecting a lehr to schedule, selecting a part to schedule on the lehr, and then scheduling additional parts along with the selected parts in the same reservation-group so that no machine setups are required. A Search Control Manager iteratively invokes each subproblem and conducts the search for scheduling alternatives according to the criteria set by the subproblems. The algorithm can be specified as:

(1) *GenerateList*. Produce *PartsList* by sorting downloaded Inventory File according to inventory level and stockout cost.
(2) *SelectNextLehr(L)*. Choose the lehr L with the earliest available start time as the lehr to focus on next for scheduling.
(3) *SelectDrivingWindshield(ws)*. From the *PartsList*, get the next unscheduled windshield *ws* as the driving part for scheduling lehr L.
(4) *SelectFamilyGroup(ws, F)*. Identify the set of parts F that can be run with the same settings and the same set of irons (i.e., racks) based on the properties of *ws*.
(5) *Schedule(ws, F)*. Try to-schedule the driving part *ws* as well as all of F, guided by the component economic run lengths.
(6) *if* Step (5) fails *then SwapParts(ws, ws_{next})*, GOTO Step (4) *else* Add the resulting reservation-group to lehr L, GOTO Step (2).

The Generative Scheduler starts the algorithm by generating an ordered PartsList from a two-key sort based on the weekly-supply and reorder point of an inventory file download from the firm's primary database. The weekly-supply shows the inventory level and possible stock-outs, whereas the reorder point reflects the batch size and the popularity of the part. The downloaded file is generated from the master production schedule by the material planning department. The Generative Scheduler chooses the earliest available lehr to schedule and then finds a part to schedule on that lehr. Since one of the most important objectives is to minimize stock-outs, a heuristic is used which selects the part with the highest priority cost. Once the windshield is chosen, it is called the "driving-windshield" since this windshield basically drives the formation of its reservation-group. Forming a reservation-group involves the scheduling of several compatible parts sequentially in time which collectively require only a single, initial setup. Since the number of windshields to be scheduled is quite large (i.e., hundreds of parts and increasing in size every year), windshields with high weekly-supplies are not included to prevent the formation of too large a group. The search for compatible parts in the formation of a reservation-group may also lead to an excessively large grouping. A stopping rule (determined by interaction with the expert) is applied in order to avoid both situations.

Given a driving-windshield, reservation-groups are formed by selecting parts that are "compatible." There are three types of parts that are compatible: parts from the same family-group (i.e., they can be processed with the same iron/rack but require different composite material combinations), parts that can be processed in the same lehr setting (i.e., they use different irons, but rely on the same oven temperatures and processing speed), and twin-parts (i.e., parts that must be processed in pairs, typically large windshields for buses, recreational vehicles, or trucks, that are composed of separately produced left and right components). Compatible parts from the same family-group are scheduled to run either sequentially (when one part type batch is completed, the next part type batch is run) or in parallel (two part types are run through the machine in the same batch) in a time interval (as constraints permit) and with only one machine

set-up required for the entire group. In forming a reservation-group, two heuristics are available. The first heuristic applied looks for a compatible part with the lowest weekly-supply and forms the reservation-group. A "cut-or-patch" procedure is used to form the group. If this heuristic fails, a second one looks for the "best fit" to form a reservation-group in order to minimize machine idle time, then the "cut-or-patch" procedure is applied.

Once the reservation-group is formed, it is applied to the schedule at the appropriate time on the appropriate lehr. If there are difficulties encountered (i.e., constraint violations), heuristics (actually, Reactive Scheduling operators) are applied to try to resolve the violations. If these adjustments cannot resolve the problem, then the driving windshield is removed from the schedule (its affiliated reservation-group is undone), it is returned to its appropriate place on the Parts-List, and the next part in the Parts-List (which has not been tried yet) is selected as the driving windshield to be applied to the schedule. If the newly selected part succeeds, then the originally selected part is the next one to be tried (as it is back on its original place in the Parts-List). Thus far, most schedules we have encountered have required only one or two such "part swaps" to solve the problem.

Constraint Checker. The primary function of the Constraint Checker is to check the feasibility of a given schedule or schedule fragment. It is used by both the Generative Scheduler and the Reactive Scheduler components in the Scheduling Kernel as well as being invoked by the human scheduler via the Manual Scheduler at the interface. The Constraint Checker inputs a schedule or schedule fragment and returns information on detected violations (if any): (a) the type of constraint violated, (b) the time location of the violation, (c) the lehr on which the violation occurred, and (d) the specific part that caused the violation. The human scheduler has the option of selecting specific constraints to enforce or ignore when invoking the Constraint Checker from the Manual Scheduler interface.

Reactive Scheduler. The Reactive Scheduler consistes of a set of components which facilitate the schedule generation process. When a potential reservation-group is formed by the Generative Scheduler, the Constraint Checker is called. If no violations occur, the reservation-group is assigned to the schedule. However, if violations are detected, the Reactive Scheduler is invoked to possibly resolve the conflict. The Reactive Scheduler is comprised of an embedded Search Manager and several Reactive Operators. Reactive Operators are independent of each other, so additional Operators may be added or dropped and existing ones may be modified. Each Reactive Operator inputs a schedule with a reservation-group violation and tries to resolve the conflict in its own manner. Currently, there are six Reactive Operators: (1) Insert idle time on a lehr, (2) Insert a reservation-group, (3) Delete a reservation-group, (4) Swap parts within a specific reservation-group, (5) Swap two reservation-groups (on the same lehr or across lehrs), and (6) Regenerate a schedule from a particular time. In the present version, the Search Manager stops when the first Reactive Operator resolves the conflict, but testing is being conducted to examine the effects of alternative search strategies and Reactive Operators on the efficiency and effectiveness of reactive scheduling.

3.2 Manual scheduler

The Manual Scheduler is the unifying structure within which all of the components may operate. The Manual Scheduler supplies both form and function consistent with the methods and perspectives of the human, but supportive of the underlying computational search strategies provided. This permits a distribution of effort between the machine and the human in a manner that permits both agents to engage in cooperative problem solving. Although the machine may be directed to "generate a schedule" on its own, it is possible for the human to augment the process in an iterative manner in which both agents can initiate problem solving, thus allowing a mixed initiative approach between the human and the machine.

Interface and manipulation routines. When running the Manual Scheduler, the user has a variety of options which effect scheduling procedures or simply review scheduling results. A summary of the options can be given as:

- Choose the number of days to schedule beginning at an arbitrary initial date.
- Invoke the Generative Scheduler (perhaps with a list of priority parts to schedule first).
- Add, Delete, or Modify scheduled parts by clicking and dragging part icons.
- Select a schedule (or portion) to have checked by the Constraint Checker.
- Manually alter the preference matrix which gives priority/stockout tradeoffs prior to generating a schedule.
- Upload/Download data and schedules from the IBM mainframe.
- Examine a variety of output reports from Generative Scheduler (e.g., load analysis, critique report).
- Add/delete/modify setup times to the schedule.
- Select parts (i.e., scheduled jobs) from an already generated schedule to save as high-priority items to schedule first in the next iteration (their position on the new schedule may vary).
- Select parts (i.e., scheduled jobs) from an already generated schedule to save at their exact position and invoke the scheduler to "schedule around" them (their position on the new schedule is fixed).

At the interface, a schedule is represented by a Gantt Chart, which is a time line for each machine on which rectangular icons depict specific jobs. The length of a rectangle reflects the duration of the job while the height of the rectangle indicates the number of parts running in the lehr. Fig. 4 presents a schedule screen from MacMerl.

When a job (i.e., rectangle) is "clicked on" by the user, it changes color and all of the relevant information regarding that particular job is displayed at the bottom of the screen. The user may choose the "pin" icon to select individual or multiple parts within or across lehrs (1) to check against the Critiquer, (2) to check against the Constraint Checker, or (3) to save in a file as a priority set of parts that must be scheduled first, and then reinvoke the Generative Scheduler to produce a new, slightly modified, schedule. The user can drag and view the generated schedule to any point in the time line of the schedule.

Critiquer. The goal of the Critiquer is to generate a critique of a schedule proposed by a human or

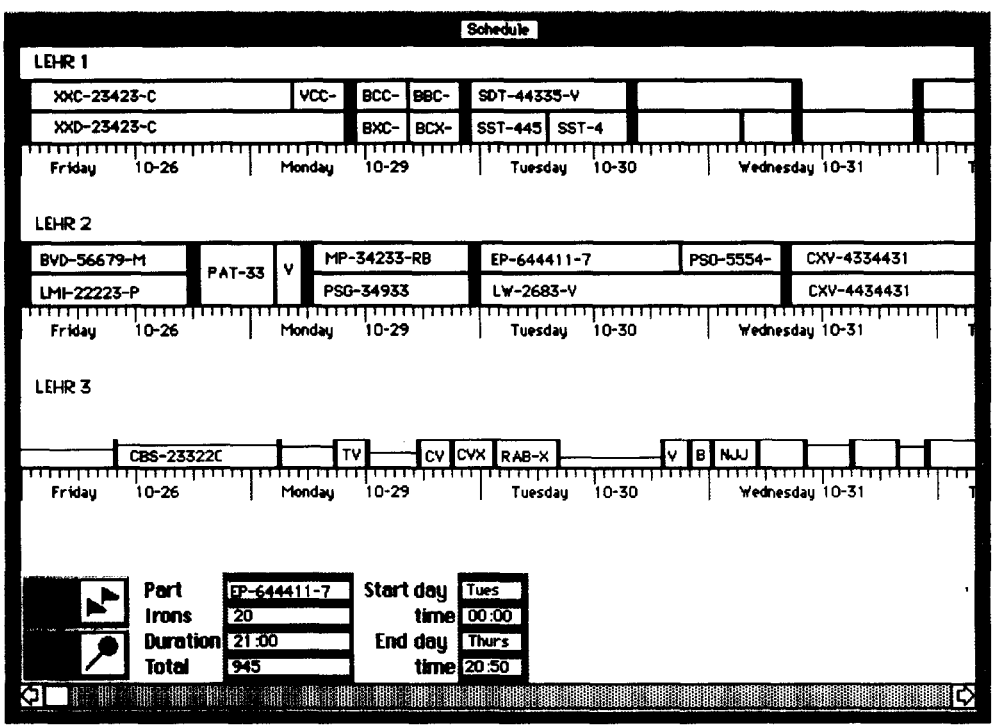


Fig. 4. A (Gantt chart) screen from MacMerl.

by MacMerl's Generative Scheduler. The Critiquer does not attempt to simulate an expert's decision making processes in order to provide good answers; rather, the Critiquer seeks to review a plan of action (in this case, a schedule) and present a concise discussion of its important, and possibly erroneous, aspects. Originally developed to critique anesthetic management plans in

```
THE FOLLOWING IS A LIST OF UNSCHEDULED LARGE PARTS WHOSE WEEK SUPPLY
IS LESS THAN 5.3 WEEKS:

RD0605V435K  WEEK SUPPLY = 4.300000
FF-1278T335E  WEEK SUPPLY = 4.400000
RD0564T190K  WEEK SUPPLY = 4.400000
RD0583V345K  WEEK SUPPLY = 4.500000
•
•
•

THE FOLLOWING IS A LIST OF UNSCHEDULED SMALL PARTS WHOSE WEEK SUPPLY
IS LESS THAN 5.3 WEEKS:


THE FOLLOWING IS A LIST OF LARGE PARTS THAT HAVE BEEN
SCHEDULED BACK TO BACK.

NOTE: CURRENTLY, 'LARGE' IS DEFINED AS HAVING WIDTH > 3200.

LEHR 2 ER445G185K / FYRR454185K ON TIME 2044 DATE 309 YEAR 1990
•
•
•


THE FOLLOWING IS A LIST OF THE STOCKOUT COSTS FOR EACH
PART SCHEDULED UP TO WEEK 5.

LEHR 1 ERT54DDDE95C  WEEK SCHED 1  WEEK SUPPLY 0.0 H COST=1
LEHR 1 ERTGDDT195_  WEEK SCHED 1  WEEK SUPPLY 0.0 G COST=2
LEHR 1 YTV554FC34R5_  WEEK SCHED 1  WEEK SUPPLY 0.9 H COST=1
•
•
•


SUM COST FOR WEEK 1 OF LEHR 1 = 42
SUM COST FOR WEEK 2 OF LEHR 1 = 41
SUM COST FOR WEEK 3 OF LEHR 1 = 49
SUM COST FOR WEEK 4 OF LEHR 1 = 24
SUM COST FOR WEEK 5 OF LEHR 1 = 0

TOTAL COST FOR FIRST 5 WEEKS OF LEHR 1 = 156
AVERAGE WEEKLY COST FOR LEHR 1 = 31.200000

SUM COST FOR WEEK 1 OF LEHR 2 = 17
SUM COST FOR WEEK 2 OF LEHR 2 = 38
SUM COST FOR WEEK 3 OF LEHR 2 = 61
SUM COST FOR WEEK 4 OF LEHR 2 = 39
SUM COST FOR WEEK 5 OF LEHR 2 = 0

TOTAL COST FOR FIRST 5 WEEKS OF LEHR 2 = 155
AVERAGE WEEKLY COST FOR LEHR 2 = 31.000000

SUM COST FOR WEEK 1 OF LEHR 3 = 68
SUM COST FOR WEEK 2 OF LEHR 3 = 20
SUM COST FOR WEEK 3 OF LEHR 3 = 0
SUM COST FOR WEEK 4 OF LEHR 3 = 0
SUM COST FOR WEEK 5 OF LEHR 3 = 0

TOTAL COST FOR FIRST 5 WEEKS OF LEHR 3 = 88
AVERAGE WEEKLY COST FOR LEHR 3 = 17.600000

TOTAL COST FOR FIRST 5 WEEKS OF ALL LEHRS = 399
AVERAGE WEEKLY COST FOR ALL LEHRS = 79.800000

CRITIQUER VERSION 1.0

!!!
```

Fig. 5. Partial report from the critiquer.

medicine [11,12], the model has also been applied in many forms. The unique nature of all these applications is the significant amount of subjective judgement involved. As a consequence, the advice suggested by the Critiquer may be appropriate, or it may not, given (the perhaps unique or unspecifiable) exogenous parameters relevant to a particular scheduling context. For example, the Critiquer checks (1) to see if the selected constraints are satisfied, (2) to see how many preferences are satisfied, and (3) how many high-priority jobs have not been scheduled. An example (partial) report of a critique is shown in fig. 5.

A schedule that has been produced by the Scheduling Kernel may subsequently be modified by the human based on additional information or, perhaps, simply the scheduler's interest in testing alternative configurations (e.g., lehr loadings, product mixes). In addition, the scheduler may wish to specify and examine partially specified schedules. The Critiquer supports the ensuing analyses of such schedule modifications/specifications by permitting multiple perspectives to be brought to bear on the schedule. These perspectives are defined by allowing or disallowing the testing of classes of constraints or preferences. Essentially, the user simply selects a schedule, or indicates a specific part of a schedule, and invokes the Critiquer. When MacMerl has generated a schedule on its own, the Critiquer is automatically invoked and the resulting report is available for review.

5. Example

We illustrate some of the capabilities of MacMerl with an example. (The nature of some of the firm data has been modified for confidentiality of product and process.) A typical weekly input for scheduling to MacMerl involves over 200 different parts, where each part has associated static data (e.g., popularity of part, properties of composite material) and dynamic data (e.g., projected demand, reorder quantities). The first step in generating a schedule is communication of the relevant constraints and optimization goals to MacMerl. Fig. 6 shows the primary screen where these options are presented.

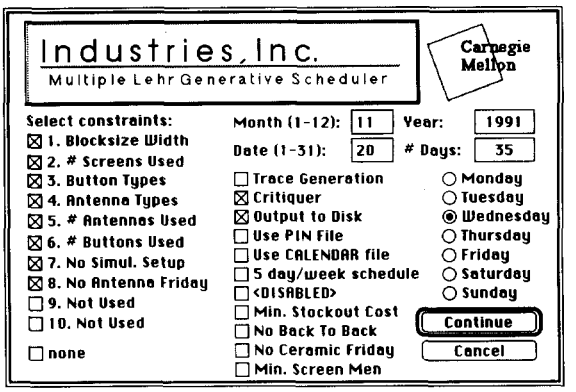


Fig. 6. MacMerl constraint selection screen.

The upper left of the screen in fig. 6 provides a series of click-boxes that indicate the eight constraints that the user may elected to have MacMerl enforce or ignore. The lower left set of click-boxes are additional "preferences" that MacMerl will try to enforce when activated. The most notable of these latter options is the stock-out cost option. When this is selected, MacMerl will use the Preference Table settings to drive a comparative analysis of tradeoffs taking stock-out costs into consideration – without this option, MacMerl will simply try to minimize stock-outs where the stock-out costs of all parts are considered equal. Fig. 7 shows the mechanism by which the user may impose comparative stock-out costs in terms of trading off the "popularity" of the part (similar to demand) with the amount of supply remaining (the entries depicted are for illustrative purposes only and do not reflect actual estimates).

MacMerl generated four schedules under four

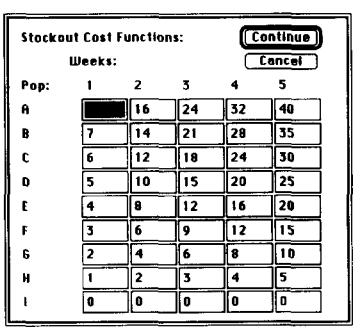


Fig. 7. MacMerl screen for imposing non-equivalent stock-out costs.

Table 1
Results of experiment examining constrait/stock-out cost sensitivities.

	Preferences			
	None		Min stock outs	
	Constraints		Constraints	
	None	All	None	All
Time (seconds)	14.2	22.4	20.2	27.9
Days scheduled	38	39	37	37
Stockout costs:				
Week 1	41	46	8	11
Week 2	39	38	47	26
Week 3	56	37	27	39
Week 4	17	19	26	31
Week 5	0	0	0	0
Total	152	140	108	107
Average	30.4	28.0	21.6	21.4

conditions with the same initial data. First, all constraints and options were inactivated. Second, all constraints were activated, but no options were selected. Third, all constraints were inactivated and one preference option was selected – minimize stock-out cost. Finally, all eight constraints were activated with the minimize stock-out cost option. The results of the computations are shown in Table 1.

The first observation we can make is that the minimization of stock-out cost effectively reduces the costs by as much as 82% in the first week and by 44% in the second with the relative improvements decreasing after that. For this firm, the most important weeks in the schedule are the first two weeks as the ordering of materials and scheduling of workers is directly related to that time frame. Subsequent weeks serve "prescheduling" functions in a rolling horizon approach that permits adjustments as required for weeks three through five. As a consequence, the improvements in the first two weeks are of most interest to the firm and it is in these weeks that the improvements are found. By not enforcing constraints, the quickest timings are generated, but the constraint violations may seriously invalidate a schedule. However, constraints in this environment can be "broken" by the human scheduler (i.e., violated under certain circumstances) and deselected for MacMerl. This example illustrates how even the enforcement of all constraints does not accrue a huge penalty in computation cost.

6. Conclusion

In this paper we have presented an overview to a system called MacMerl, which provides a set of computer-based scheduling tools for the user. The primary design of MacMerl was driven both by a detailed analysis of the task and a detailed analysis of an expert's adaptation to the task. The examination of the expert's strategies not only provided insight into the design of the interface, but suggested interesting ways to generate and modify schedules. By augmenting these strategies with algorithmic approaches, schedules that once took hours to generate now take seconds. Furthermore, the manual method of scheduling only addressed a 5-week horizon MacMerl can generate a 39 day schedule in seconds. As a result, the system serves as an intelligent scheduling colleague that can suggest and accept suggestions regarding the scheduling process (e.g., enforce or ignore constraints and preferences) as well as the schedule itself (e.g., schedule a selected set of jobs first). Thus, the human and the machine cooperate to form a mixed-initiative scheduling environment. Furthermore, upper management is now interested in the system in order to study projections and schedule attributes when modifying the various stockout costs and relative importance of parts. The schedules generated by MacMerl are not optimal; however, they are currently "as good as" those generated by the expert and sometimes exceed the expert's. Given that such schedules can be generated in a short period of time, they serve to incrementally increase the value of the task for the firm. This is significant as the expert will not always be there so there must be some way to capture and extend the expert's capabilities. Finally, MacMerl is serving as an interesting context for developing and testing alternative search strategies and heuristics within an actual, complex, task environment. For example, we are currently experimenting with applying beam search [13] and other search strategies to the task of selecting driving windshields. Additionally, our efforts are also focusing on defining and linking a sufficient set of reactive

scheduling operators in order to eventually totally automate the reactive scheduling task itself.

References

[1] R. Bonczek, C. Holsapple and A. Whinston, Foundations of Decision Support Systems (Academic Press, New York, 1981).

[2] G. Bruno, A. Elia and P. Laface, A Rule-Based System to Schedule Production, IEEE Computer (July 1986) 32–40.

[3] M. Fox and S. Smith, ISIS: A Knowledge-Based System for Factory Scheduling, Expert Systems 1, No. 1 (1984) 25–49.

[4] R. Graham, The Combinatorial Mathematics of Scheduling, Scientific American (March 1978) 124–132.

[5] S. Graves, A Review of Production Scheduling, Operations Research 29, No. 4 (1981) 646–675.

[6] A. Hax and D. Candea, Production and Inventory Management (Prentice-Hall, Englewood Cliffs, NJ, 1984).

[7] B. Huguenard, M. Prietula and F.J. Lerch, Performance ≠ Behavior: A Study in the Fragility of Expertise, Proceedings of the 10th International Conference on Information Systems (Boston, MA, 1989).

[8] J. Kowalik, Ed., Coupling Symbolic and Numeric Computing in Expert Systems (North-Holland New, York, 1986).

[9] K. McKay, J. Buzacott and F. Safayeni, The Scheduler's Knowledge of Uncertainty: The Missing Link, Paper presented at the IFIP Working Conference on Knowledge-based Production Mangement (Galway, Ireland, Aug. 1988).

[10] K. McKay, F. Safayeni and J. Buzacott, Job-Shop Scheduling Theory – What is relevant? Interfaces 18, No. 4 (1988) 84–90.

[11] P. Miller, A Critiquing Approach to Expert Computer Advice: ATTENDING (Pitman, London, Boston, 1984).

[12] P. Miller, Expert Critiquing Systems. (Springer-Verlag New York 1986).

[13] P.S. Ow and T.E. Morton, Filtered Beam Search in Scheduling, International Journal of Production Research 26, No. 1 (1988) 35–62.

[14] P.S. Ow and S. Smith, Viewing Scheduling as an Opportunistic Problem-Solving Process, Annals of Operations Research 12 (1988) 85–108.

[15] M. Prietula and H. Simon, The Experts in Your Midst, Harvard Business Review (Jan.–Feb. 1989) 120–124.

[16] S. Smith, M. Fox and P.S. Ow, Constructing and Maintaining Detailed Production Plans: Investigations into the development of Knowledge-Based Factory Scheduling Systems, AI Magazine (Fall 1986) 45–61.