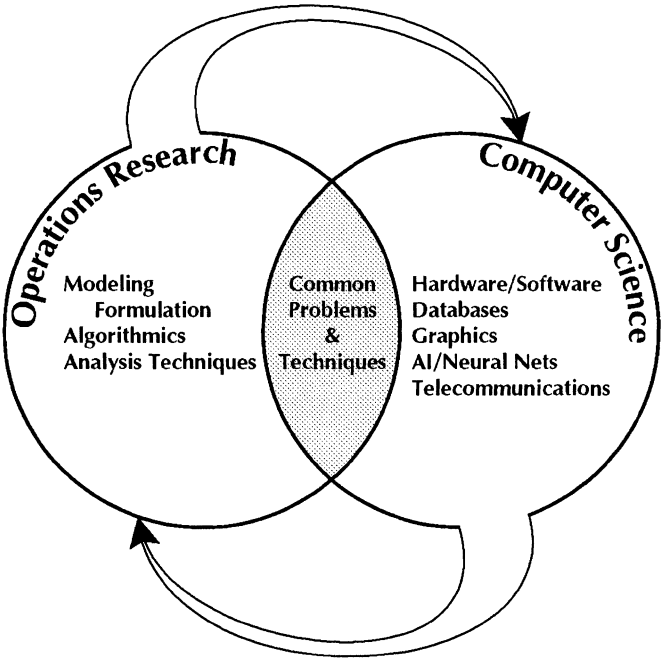
Intelligent Scheduling Systems

OPERATIONS RESEARCH / COMPUTER SCIENCE INTERFACES SERIES



Series Editor

Ramesh Sharda
Conoco/Dupont Chair of Management of Technology
Oklahoma State University
Stillwater, Oklahoma U.S.A.

Other published titles in the series:

Greenberg, Harvey J.
University of Colorado @ Denver
A Computer-Assisted Analysis System for Mathematical Programming Models and Solutions: A User's Guide for ANALYZE©

Greenberg, Harvey J.
University of Colorado @ Denver
Modeling by Object-Driven Linear Elemental Relations: A Users Guide for MODLER©

Intelligent Scheduling Systems

edited by
Donald E. Brown
William T. Scherer
University of Virginia

Springer Science+Business Media, LLC

Library of Congress Cataloging-in-Publication Data

Intelligent scheduling systems / edited by Donald E. Brown, William T. Scherer
p. cm. -- (Operations research/computer science interface series)
Papers presented at the Intelligent Scheduling Systems Symposium, sponsored by the Artificial Intelligence Technical Section of the Operations Research Society of America.
Includes bibliographical references and index.
ISBN 978-1-4613-5954-8 ISBN 978-1-4615-2263-8 (eBook)
DOI 10.1007/978-1-4615-2263-8
1. Scheduling (Management) 2. Scheduling (Management)--Data processing. I. Brown, Donald E. II. Scherer, William T. III. Intelligent Scheduling Systems Symposium. IV. Operations Research Society of America. Artificial Intelligence Technical Section. V. Series.
TS157.5.I58 1994
658.5'3--dc20 94-36169
CIP

Contents

Contributors

Professor J. Wesley Barnes, Cullen Trust for Higher Education Endowed Professor in Engineering, Graduate Program in Operations Research and Industrial Engineering, Department of Mechanical Engineering, ETC 5.128D, The University of Texas at Austin, Austin, Texas 78712

Professor E. Andrew Boyd, Department of Industrial Engineering, Texas A&M University, 238 Zachry Building, College Station, Texas 77843-3131

Professor Rusty Burlingame, Department of Industrial Engineering, Texas A&M University 238 Zachry Building, College Station, Texas 77843-3131

Professor John Buzacott, Faculty of Administrative Studies, York University, Downsview, Ontario, Canada M3J 1P3

Dr. Ken Darby-Dowman, Department of Mathematics and Statistics, Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K.

Raymond Fink, Idaho National Engineering Laboratory, EG&G Idaho, Inc. P.O. Box 1625, Idaho Falls, Idaho 83415-2407

Kevin Gary, Global Associates Ltd., Arlington, Virginia

Professor Fred Glover, U S. West Chair in Systems Science, Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, Colorado 80309-0419

Dr. Othar Hansson, Heuristicrats Research, Inc. 1678 Shattuck Avenue, Suite 310, Berkeley, California 94709

Dr. Karl Kempf, Knowledge Applications Laboratory, Intel Corporation, Chandler, Arizona 85226

Leonard Kingsley, United States Coast Guard R & D Center, Avery Point, Groton, Connecticut 06340-6096

Professor Manuel Laguna, Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, Colorado 80309-0419

Dr. Kenneth S. Lindsay, The MITRE Corporation, Center for Advanced Aviation System Development, 7525 Colshire Drive, McLean, Virginia 22102-3481

Dr. Cormac Lucas, Department of Mathematics and Statistics, Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K.

Professor Andrew Mayer, Computer Science Division, 571 Evans Hall University of California Berkeley, California 94720

Professor Kenneth McKay, Faculty of Business Administration, Memorial University of Newfoundland, St. John's, Newfoundland, A1B 3X5, Canada

Professor Gautam Mitra, Department of Mathematics and Statistics, Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K.

Professor Thomas E. Morton, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213

Professor Sang Chan Park, Graduate School of Business, University of Wisconsin, Madison, Wisconsin 53706

Professor Selwyn Piramuthu, Department of Decision and Information Science, University of Florida, Gainesville, Florida 32611

Professor Narayan Raman, Department of Business Administration, University of Illinois, Champaign, Illinois 61820

Professor Prasad Ramnath, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213

Professor Frank Safayeni, Dept. of Management Sciences, University of Waterloo,Waterloo, Ontario, CANADA N2K 2G4

Professor Michael J. Shaw, Department of Business Administration, University of Illinois, Champaign, Illinois 61820 and Beckman Institute, University of Illinois, Urbana, Illinois 61801

Professor Stephen P. Smith, Center for Integrated Manufacturing Decision Systems, the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213

J. Walter Smith, (1938-1993)

Professor Reha Uzsoy, School of Industrial Engineering, 1287 Grissom Hall, Purdue University, West Lafayette, Indiana 47907-1287

Preface

This book provides current results from research in scheduling using methods from operations research (OR) and artificial intelligence (AI). Scheduling is a resource allocation problem which exists in virtually every type of organization. The ubiquity of scheduling problems has produced roughly 40 years of research. Within the OR community this research has emphasized mathematical modeling techniques which seek exact solutions to well- formulated optimization problems. While this approach has produced important results, particularly in continuous flow production, most real-world scheduling problems seem resistant to solution with these methods.

Hence, over the last decade operations researchers and management scientists interested in scheduling have turned increasingly to more computer intensive and heuristic approaches. At roughly the same time, researchers in AI began to focus their methods on industrial and management science applications as a way to move beyond the toy problems typical of AI research in the 60's. The result of this confluence of fields has been a period of remarkable growth and excitement for workers in scheduling. This book seeks to capture results which are at the forefront of this new wave of scheduling research.

This book will be of interest to both researchers and practitioners working in scheduling. Practitioners will benefit from the broad coverage in the book. Papers are provided in the two major application areas: production and transportation scheduling. In addition, the first part of the book, Issues in Scheduling, provides an overview of the current work in scheduling and important issues under investigation. Practitioners will also benefit from the book's presentation of a variety of contemporary tools to assist in operational scheduling.

Researchers will find the blending of techniques from AI and OR for complex problem solving to be of interest. Real scheduling problems are typically among the most difficult problems known. The results here show how many of the latest techniques from tabu search to genetic algorithms can provide effective solutions to these difficult problems. Finally, researchers will find a wealth of pointers to challenging new problems and potential opportunities that exist in scheduling.

In summary readers of this book will benefit from it in the following ways:

1. Explore a wide range of scheduling applications;
2. Develop insight into the complexity of scheduling problems and the latest techniques for their solution;

3. Discover how results from the fields of OR and AI are addressing one of the most widespread resource allocation problems; and
4. Find pointers to new research areas with references from workers at the forefront of the field.

The papers in this book were presented at the Intelligent Scheduling Systems Symposium, sponsored by the Artificial Intelligence Technical Section (AITS) of the Operations Research Society of America. Officers and council members AITS wanted to provide a forum for the exchange of ideas in scheduling. The symposium was a major success bringing together an excellent group of researchers and practitioners. The papers chosen for this book were selected from the symposium and subject to a review process designed to yield the papers with important contributions to scheduling theory and/or practice.

We could not have completed this book without the considerable assistance of others. In particular, we acknowledge the support received in the preparation of this work from the officers and members of the AITS. The project was particularly encouraged by one of the first Chairs of AITS, Jerry May. In addition, the symposium and this book owe a special debt to Laura Davis who was AITS Chair during the symposium. Time and again Laura helped with the organization of the symposium and with financial details. We also appreciate the support of the referees, who assisted us in reviewing the submitted papers. The sister organization of AITS, the Institute of Management Sciences College on Artificial Intelligence also supported the project and we received the excellent advice of their president, Jay Liebowitz. Jack Marin, at the University of Virginia, also provided extra help in the final stages of putting the book together. Finally, we owe special thanks to Jeanette Davis, who assisted us throughout the preparation of the book: maintaining files, writing letters, calling authors, reviewing formats, and generally helping with the completion of the book.

Donald E. Brown
William T. Scherer

Charlottesville, Virginia

A Survey of Intelligent Scheduling Systems

DONALD E. BROWN*, JOHN A. MARIN*, WILLIAM T. SCHERER*

* Department of Systems Engineering, University of Virginia

Abstract

Scheduling involves the effective allocation of resources over time. In this paper, we examine the impact of intelligent systems on production, transportation, and project scheduling. We provide a survey of intelligent scheduling systems based on artificial and computational intelligence techniques. These methods include knowledge-based systems, expert systems, genetic algorithms, simulated annealing, neural networks, and hybrid systems. We also review the basic components of the scheduling problem and describe some potentially promising areas for future research.

key words scheduling, artificial intelligence, computational intelligence, project management

1. Introduction

Scheduling concerns the effective allocation of resources over time. Most organizations, both large and small, must solve scheduling problems on a recurrent basis and this creates considerable demand for good scheduling techniques. Starting in the 1950's operations researchers began advocating formal optimization approaches to scheduling. Unfortunately, after four decades of work these methods can still only guarantee optimality for a very limited set of problems. Simon [1] states that

> ...while OR [operations research] has been able to provide a theoretical analysis of scheduling problems for very simple situations — a job shop with a very small number of machines and processes — automatic scheduling of large shops with many machines and processes has been beyond the scope of exact optimization methods. [p. 13]

To reinforce Simon's claim, considerable evidence exists that operations research techniques have not, for the most part, replaced the manual trial-and-error process employed by human schedulers [2, 3, 4].

One reason optimization methods fail is because most scheduling problems belong to the class of NP-Hard problems [5, 6]. These problems are distinguished by rapid growth in the number of potential solutions with modest growth in the number of resources to be scheduled. This growth is so rapid that even the fastest computer imaginable could not search through every potential solution to real problems in a reasonable amount of time. Further, no clever *general-purpose* approach to pruning the search space is known. Hence, if we want to improve scheduling in real operations we must turn to other methods.

Another reason for the limited success of traditional optimization approaches to scheduling is the difficulty in capturing the problem formulation in a closed form mathematical expression. Most real-world problems have numerous complications that range from differential labor pricing to vaguely stated and competing objectives. In some scheduling problems the proposed schedule can only be evaluated by simulation, which means both increased computational expense and uncertainty in the interpretation of the results. This difficulty in matching the real scheduling problem to the single, closed-form objective functions found in traditional optimization approaches is perhaps the major reason why most scheduling is still done manually.

Because of these difficulties many researchers in scheduling have turned to techniques from artificial and computational intelligence (e.g. Simon [1]). Artificial intelligence (AI) uses symbol-processing computer programs to strive for human-like (or better) performance. AI emphasizes symbolic methods and

grounds this focus on the yet unproved symbol systems hypothesis [7]: computer programs which manipulate symbols are fully capable of intelligent (human) behavior. Among all the areas in AI, knowledge-based (or expert) systems [8] have had the broadest range of success and applicability. More recently, machine learning methods, such as rule induction, have also proven valuable in scheduling operations.

In contrast to AI, computational intelligence (CI) uses numerical methods for modeling human and, more generally, biological performance in problem solving. Whereas AI employs symbolic and conceptual structures of the mind as the building blocks for intelligence, CI starts with models of the neuron or other basic building blocks. Neural networks are the most commonly recognized CI technique, but others with applicability to scheduling include genetic algorithms [9], evolutionary programming [10], simulated annealing [11, 12], and inductive learning networks [13]. All of these techniques are examples of self-organizing systems which form a major subgroup of CI. Both simulated annealing and genetic algorithms are frequently classified as heuristic algorithms, but we prefer to consider heuristics as a way to aid problem solving by specifically accounting for domain information. In contrast, genetic algorithms, simulated annealing, and other inductive methods follow a general computational paradigm to "learn" promising directions for search. Thus, where AI has roots in cognitive psychology, CI has its roots in neurophysiology and the self-organizing paradigms of cybernetics.

The fields of OR, AI, and CI merge in the area of heuristic search. AI has a tradition of exploiting human-like problem solving knowledge which can sometimes be expressed as heuristics. In AI the challenge has been to encode and control this knowledge. When included in a formal framework, such as the graph search formalism of A* [14], these heuristics can provide scheduling procedures with guaranteed optimal performance. Even when we cannot achieve optimality the combination of heuristics with a formal search method can produce exceptional results.

CI has exploited small computational building blocks to achieve intelligent behavior. However, this raw computational approach tends to overlook problem specific characteristics that could aid problem solving. The method of presenting examples in backpropagation neural networks and the preservation of diversity in genetic algorithms are examples of the use of domain knowledge to enhance performance. Hence, heuristic search can be viewed as a blending of AI, CI, and OR, which has showed particular promise for scheduling.

Unfortunately, the terminology surrounding heuristic search has tended to become confusing. The debate on the meaning of heuristics sometimes resembles a religious argument. We attempt to minimize this controversy by following concepts from the theory of algorithms. In particular, exact

algorithms carry a guarantee of finding an optimal solution, if such a solution exists. Inexact algorithms have no such guarantee. However, in most cases they have been shown (primarily through experience) to yield good solutions. We use the term heuristics to describe problem solving knowledge, which is typically problem specific.

Because of the operational variety and inherent difficulty of most scheduling problems, techniques and combinations of techniques from AI, CI, and OR have a place in the toolkit of both the scheduling practitioner and researcher. We use the unabashedly exaggerated rubric, "intelligent systems" to signify the collection of these approaches. In this paper we survey representative methods from intelligent systems for scheduling. Other surveys have looked at the scheduling literature in specific application domains, while here we examine broadly scheduling systems across the major applications and across intelligent systems techniques.

This double focus means we could organize our survey around either applications or techniques. We chose applications at the highest level and techniques second, selecting this organization because of the widespread recognition that universal approaches to scheduling are unrealistic. While both practitioner and researcher would like a general purpose scheduler that generates optimal schedules across all application domains, real scheduling has proven particularly immune to this type of general approach. Instead, tailored procedures which employ domain specific information have proven most effective. Even general purpose optimization procedures work best when tailored to the actual scheduling problem. This means that scheduling researchers and practitioners must concern themselves with the details of the application domain in addition to the techniques employed.

Three application areas appear widely in the scheduling literature: production, transportation, and project. The first two typically involve operations and recurrent decision making, while the third concerns planning and design. In each of the sections on these domains we begin with a brief examination of the salient problem features and provide entry points to the OR literature on exact and inexact methods. We then provide subsections on scheduling work from AI and CI, respectively. The literature in scheduling is enormous and we make no claims of completeness in this survey. Instead we have attempted to provide pointers to representative techniques from intelligent scheduling systems. We conclude this section with a summary section which also provides directions to potentially promising areas of research within the field of intelligent scheduling systems.

2. Production Scheduling Problems

Production scheduling involves resource allocation over time for the manufacture of goods. Typical resources of concern to manufacturing are labor, materials, and equipment which are allocated to minimize costs. Production scheduling problems frequently have constraints such as processing time, sequencing order, availability of resources, and demand.

In a general production scheduling problem, there exists a set of n jobs to be processed through m machines. The process of a job on a machine is called an operation. Each operation takes a certain length of time to perform, and each job has a ready time or release date.

Products typically require multiple operations. Hence, the scheduling problem must account for the sequencing of jobs on machines. The desire to run multiple products through the machines, where each product requires a different job sequence is a problem of immense importance to contemporary manufacturing.

Real production scheduling problems are also dynamic in nature. Graves [15] states, "A frequent comment heard in many scheduling shops is that there is no scheduling problem but rather a rescheduling problem."[p. 646] Hadavi, et al. [16] describe the dynamic nature of production scheduling, declaring,

> Machines breakdown, product quality assurance, supplier delivery, operator availability, and continual arrival of new orders can all be part of the dynamics of the environment . Decision making for scheduling can also occur on many different levels: weekly to quarterly capacity planning; daily shift-level detailed scheduling; and on-the-spot reactions to machine breakdowns, shortage of supplies and so on. [p. 46]

The importance of production scheduling has lead to a variety of approaches to different aspects of this application. We can conveniently group the approaches along the three problem dimensions suggested by Graves [15]:

1. Source of Requirements— requirements can come either directly from customer orders or indirectly from inventory availability. This distinction is sometimes termed open versus closed shop.
2. Processing Complexity — delineates the number and extent of the processing tasks associated with each operation. A common breakdown includes:

a. One stage, one processor;
b. One stage, parallel processor;
c. Multi-stage, flow shop;
d. Multi-stage, job shop.

3. Scheduling Criteria — indicates the performance measures used to evaluate schedules. Examples of performance measures include production time, idle time, due dates, maximum lateness, mean flow time, inventory costs, and utilization costs. The goals to maximize or minimize these performance measures sometimes conflict.. "For example, the urge to meet due dates encourages the scheduler to start jobs as early as possible. However, to minimize the work in progress and finished goods inventories, the scheduler attempts to postpone starting the job as long as possible [16, p. 46].

Production scheduling has been an area of interest to the OR community dating to the founding of the field. Hence, there exists a large literature on exact and inexact algorithms for production scheduling. The books of Baker [17] and French [18] provide good descriptions of these techniques and the surveys by Rodammer and White [19] and White [20] give insight into the contemporary work. Despite the quantity and quality of the OR work in this area, many outstanding issues preclude the use of this research for practical applications. It is these issues that have driven many to explore AI and CI approaches for production scheduling, which we review in the next two subsections. However, before doing so, we note several interesting developments in inexact algorithms and heuristic search that relate to directly to work in intelligent systems. The first concerns distributed scheduling systems and the second involves the use of a newer search method – tabu search.

Since production scheduling decisions in large shops are not made by any single individual, distributed scheduling architectures are needed. Ow et al. [21] developed a distributed system, the Cooperative Scheduling System (CSS) to manage this problem. CSS both estimates completion times and schedules work-orders. An interesting element of this design is the evaluation of objectives by each resource broker. In many systems, one centralized component evaluates perspective trade-offs, however, in CSS several sub-modules concurrently evaluate proposals for the best solution.

Hadavi, et al. [16] describe a real-time distributed scheduling system named REDS. The fundamental structure used for representing constraints and schedules is a temporal tree, which has the advantage of looking at conditions in both the past and future. Each resource has its own temporal constraint tree, and the schedule is represented as a temporal schedule tree. Each node in a tree corresponds to an interval in time; thus, the children of a node form a partition based on the time interval. This representation allows REDS to expeditiously

check for possible problems, such as bottlenecks, without having to search the entire tree. REDS has been used in a large-scale integrated development factory for a printed circuit boards and in a job shop, mask manufacturing plant.

Another important development in inexact algorithms for scheduling was the introduction of tabu search [22, 23]. This method uses a concept of "memory" to avoid local optima.

Tabu search has been successfully applied to a scheduling problem for distributing workloads among machines to accomplish all processing demands over a specified time horizon. This problem was formulated as a nonlinear, generalized covering problem involving 300 to 500 integer variables and about 100 constraints. Glover reports that tabu search solved the problem with values half as large as previous methods [24].

Widmer [25] also reports good results using tabu search in a flexible manufacturing environment where set-up times for operations are not included in the processing times. The author refers to these set-up times as "tooling constraints."

Recently, Glover [26] has described ways to apply tabu search to nonlinear and parametric optimization problems. Glover defines a parametric optimization problem as problems which "... involve searching an auxiliary space in order to obtain vectors that map solutions in a primary space." [p. 232] These types of problems commonly arise in production planning and scheduling, and include the multiconstraint knapsack and job shop scheduling problems. Glover employs "algorithmic mappings" to search a nonlinear continuous space to link outcomes in different solution spaces. One approach, termed the "scatter search", uses weighted combinations of previous reference points to create a series of new reference points and may be used to augment genetic algorithms.

2.1 AI Methods for Production Scheduling

Knowledge-Based Systems (KBSs) provide the major AI approach to scheduling in general, and production scheduling in particular. KBSs employ domain specific problem solving information to derive schedules. Typically this knowledge is encoded in rules. These systems do not have nor do they aspire to a guarantee of optimality. Instead, they attempt to employ reasoning that is easily understood and accepted by human users to produce good schedules. Ignizio [27] contains a thorough discussion of KBSs and their components.

KBSs have been developed for a number of important scheduling problems. We roughly divide the literature on KBSs into those that use or do not use dispatching rules. In the latter category, we have KBSs that use machine specific knowledge, constraint maintenance, and hybrid systems.

Dispatching rules figure prominently as knowledge for use within production scheduling KBSs. Blackstone, et al. [28] define a dispatching rule as a rule "used to select the next job to be processed from a set of jobs awaiting service." An example of a simple dispatching rule is "Select the next job that has been waiting the longest" while a complicated dispatching rule is "Select the job with the shortest due date whose customer inventory is less than a specified amount."

Some prominent investigations into the use of dispatching rules include Conway and Maxwell [29], Elvers [30], Conway [31], and Rochette and Sadowski [32]. These and other dispatching rule approaches are thoroughly examined by Blackstone, et al. [28] and they formally define 34 dispatching rules and provide an analysis of these rules. [28] uses simulation to compare several dispatching rules. The paper concludes that no rule is best in all circumstances, however, some rules usually perform better than others. For example, "The shortest operation rule (SI) selects for processing that job for which the operation can be computed in the least time". [p. 33] Testing the SI rule using simulation experiments, the authors conclude that "SI appears to be the best dispatching rule except in the case where due dates are set at seven or more times the total work content and machine utilization is moderate." [p. 34] We note that this latter statement is itself a heuristic and could be included in a system to choose dispatching rules.

A KBS built by Grant [33] called FIXER uses dispatching rules (referred to as loading rules) to schedule a Royal Air Force squadron repair shop. Grant chose to build a KBS for this application because the precedence network is constantly changing in this application making traditional methods inappropriate. Since different dispatching rules are appropriate for different circumstances, the FIXER employs a forward-chaining inference engine to determine when to use the appropriate dispatching rule, given the current objective. When the system is queried by the user, the dispatching rule selected by FIXER is applied to the list of outstanding jobs. The rules used by FIXER to select the appropriate dispatching are not explicitly stated by Grant. However, it is implied these rules are directly related to the objective of the dispatching rule itself.

Pierreval and Ralambondrainy [34] built on the use of dispatching rules by introducing learning algorithms that generate a set of production rules. The authors describe a program called GENREG which determines production rules based on a set of simulation-derived observations. The simulation model is embedded in the expert system, creating a hybrid system. "The simulation model may be called whenever it is necessary to choose dispatching rules, in order to compare their performance according to the state of the manufacturing system." [p. 463]. If rules currently in the knowledge base cannot accommodate the results from the simulation, a new rule is established and added to the knowledge base. Although the authors report encouraging results, there are

some limitations to this approach. The time required to run the simulation when a new situation arises precludes the operation of this system in real-time. Also, the system is dependent upon the quality of the simulation and interpretation of ensuing results for meaningful information. Finally, methods are needed to decide when dispatching rules already present in the knowledge base are insufficient.

Systems that do not use dispatching rules have employed other knowledge and approaches for scheduling. Gulf States Steel [35] employs an expert system to plan and schedule continuous slab caster production. The knowledge-intensive system integrates data from orders with product, process, metallurgical and equipment data. The result is a system which easily and quickly generates and updates plans and schedules based on production feedback status reports.

Another popular approach to scheduling uses constraint-based reasoning. An illustration of this approach is the ISIS system developed for the Westinghouse Turbine Component Plant [36]. The ISIS system is based on the premise that "Schedule construction can be cast as a constraint directed activity that is influenced by all relevant scheduling knowledge." [p. 25] This view of constraints dictating the search procedure leads to the philosophy behind ISIS that "Constraints must be selectively relaxed and the problem-solving strategy must be one of finding a solution that best satisfies the constraints." [p. 26] This approach views scheduling from the perspective of satisfying constraints, and reduces the influence other effects such as temporal variables or criticality of certain operations.

ISIS employ frames to encode the knowledge necessary to solve a scheduling problem. Since ISIS relies on constraint-directed search, the authors state that much of their research has centered on identifying and characterizing constraints. Fox and Smith [36] categorize and explain the important aspects of constraint knowledge.

Ow and Smith [37] extended the capabilities of ISIS to develop OPIS, which combines opportunistic reasoning and hierarchical organization to solve production scheduling problems. Opportunistic reasoning means the "system constantly is on the lookout for ways to alter or develop its plan of action based on the latest information." [p. 432] The hierarchical organization distributes the planning and control functions to a group of individual knowledge sources, each supplied with its own set of heuristics.

OPIS directs its effort towards aspects of the problem with the fewest choices and/or most powerful heuristics. Thus, rather then schedule the problem as a whole, OPIS schedules sub-problems as they arise. OPIS does not exhibit the single perspective of satisfying constraints exhibited by ISIS, OPIS instead

views all operations causing a bottleneck as a critical. The current version of OPIS runs slowly, and is not appropriate for real-time use.

Another constraint-based approach employs the user to guide the scheduling effort. According to Elleby, et al. [38] their system is based on the belief that "Desirable and undesirable features of a proposed schedule can often be identified by a human user, although it is very difficult to define explicitly what constitutes a good schedule." [p. 1] The authors describe a system in which a schedule is derived, presented to a human expert for comment and critique, and then this knowledge is incorporated into the system for the generation of future schedules. The knowledge obtained from the human scheduler results in a requirement, or constraint, that the system stores for future use. The system consists of three components:

1. A constraint maintenance system;
2. A Schedule Generator, and;
3. Request Interpreter.

The authors assert that the task of the schedule generator is equivalent to solving a constraint satisfaction problem, and propose the use of an incremental constraint satisfaction routine using a backtracking approach. The backtracking approach enlarges a partial assignment of values to variables from their domain until an inconsistency or acceptable solution is found. The order in which the variables are assigned values (instantiation order) determines the amount of time this procedure consumes. While the authors report the performance of the system is "very encouraging", they do not suggest an approach to determining the instantiation order. Also, this approach is highly dependent on the expertise of the user required to comment and critique the proposed schedules.

Another representative approach to production scheduling is to combine different types of knowledge (e.g. dispatch rules, constraints) in KBSs or to combine KBSs with optimization. The former approach is exemplified in OPAL [39]. The OPAL system "uses several conceptual entities and tools in order to account for different kinds of knowledge involved in the formulation of the (scheduling) problem, description of the data, and solution methods." [p. 796] They divide the knowledge into static and dynamic knowledge.

Dynamic knowledge is the available expertise about the schedule construction. Part of the dynamic knowledge is empirical expertise, which includes the dispatching rules previously described. Static knowledge is the problem data, such as machines available. OPAL is built around a database containing the static knowledge, a constraint-based analysis module and a decision support module. OPAL is written in COMMON- LISP, with every piece of knowledge encoded as an object in an object-oriented language approach. OPAL provides good short-term scheduling solutions.

Another approach to hybrid systems combines knowledge in a KBS with search. A Knowledge-Based Scheduling System (KBSS)[40] incorporates a heuristic search algorithm into a knowledge-based system. "The algorithm includes a number of priority rules. The scheduling instances which cannot be handled by the algorithm are referred to the inference engine." [p. 454] The inference engine uses a set of production rules, not related to the dispatching rules mentioned earlier, to generate a proposed schedule. Frames are used to represent the declarative knowledge used in describing the production problem.

KBSS was evaluated in [40] on a set of test problems using three measures of performance: maximum flow time, average flow time, and machine utilization. Kusiak states the KBSS schedules are of good quality, usually within a couple percentage points of the optimal. Also, the CPU time used is modest, indicating KBSS may be suitable for real-time operations.

Finally, Binbasioglu and Jarke [41] describe an expert system for developing linear programs to solve scheduling problems. The intent of such hybrid systems is to allow non-professionals to easily employ optimization techniques. This approach clearly has promise in those instances where exact or relaxed optimization algorithms are appropriate.

2.2 CI Methods for Production Scheduling

CI methods have only recently been applied to production scheduling problems. Nonetheless the importance of production scheduling and the promise of improved performance using CI has encouraged a number of researchers to investigate methods from this area. In this section we provide examples using genetic algorithms, simulated annealing, and neural networks.

Genetic algorithms (GA) mimic the process of natural evolution in searching a solution space for near-optimal solutions. John Holland conceived the idea that a computer algorithm might solve a difficult problem the way nature does — through evolution [42]. Good introductions to GAs can be found in [42, 43].

A basic approach in the application of GAs to job shop scheduling is provided by Biegel and Davern [44]. This technique positions GAs in the scheduling process "to receive outputs from both the automated process planning function and the order entry function." [p. 81] The authors indicate positive results using GAs in this manner. Gupta et al. [45] examine the problem of scheduling n jobs on a single machine with a non-regular objective function. The objective is to minimize "the squared deviations from the mean completion time or flow time." The authors also provide an in-depth analysis of parameter selection for the GA.

One of the major problems with applying genetic algorithms to scheduling is obtaining a good problem representation that GA operators can effectively exploit. Whitley, et al. [46] describe a representation and GA crossover operator, the edge recombination operator, specifically designed for sequencing problems. They applied this operator to a production line scheduling problem modeled after a circuit board assembly line at a Hewlett-Packard factory in Fort Collins, Colorado. The authors report favorable results from their system. Additionally, the authors performed a worst case analysis of the edge recombination technique in creating hypothetical systems where recombination is suspected to fail. Results from a series of experiments show it is possible to create a scheduling problem where the edge recombination performs unfavorably. Specifically, parallel machines processing jobs from a single queue caused the GAs to perform less favorable than in previous tests.

Another GA approach by Syswerda [47] combines a deterministic scheduler to build initially feasible schedules for the GA to improve. The system provides schedules for a U.S Navy System Integration Test Station (SITS) for F-14 airplanes. The deterministic schedule builder takes a particular task schedule and builds a feasible schedule, in which the order of the tasks do not violate any known constraints. Next, GAs take a feasible schedule and turn it into a good schedule. In this model, the chromosome representation in the GA consists of orderings of tasks. Three different mutation and crossover operators were tested.

Uckun, et al. [48] created a system called the Vanderbilt Schedule Optimizer Prototype (VSOP) to explore the use of GAs in job shop scheduling. The author's state, "Our project's goals were to develop a knowledge representation formalism for job shop scheduling, explore GAs as a viable search method, and develop a prototype problem solver for proof-of-principle." [p. 35]

Chromosomes in VSOP represent a particular ordering of operations for a certain machine. This representation is quite flexible and easily adapts to the re-scheduling problem. For example, high priority orders are easily inserted into the chromosome. The authors also represent problem-specific information in the chromosome, such as a particular machine plan for producing a certain item.

GAs quickly identify possible solution areas, but may not converge to the optimal solution rapidly [48]. Therefore, VSOP also includes a "local hillclimbing" algorithm to improve upon the GAs proposed schedule. If the hillclimbing algorithm improves upon the GAs schedule, then this modified chromosome is inserted back into the GA for further evolution. The authors tested VSOP on a series of simulated job shop problems. Results for a simple GA combined with limited local search produced acceptable results. However, a GA enhanced with problem-specific information and edge recombination operators produced even better results.

A chemical flowshop is a production scheduling problem exhibiting a single set of sequential processing units (reactors, dryers, centrifuges), producing a variety of different chemicals. Cartwright and Long [49] use genetic algorithms to simultaneously schedule operations and refine the topology of a chemical flowshop. Given a set of chemicals to be processed, the number and type of processing units available, processing times, and rules regarding storage requirements, the objective usually is to minimize the total processing time (makespan). In order to provide a more flexible system, the authors allow for the placement of some additional processing units parallel to existing units. However, the addition of even a few extra processing units greatly complicates the calculation of the makespan because of the different "routes" now available. A comparative analysis using genetic algorithms found it is better to optimize the topology and feedorder together rather than optimize them separately. Also, the authors found, "A genetic algorithm provides a fast and reliable method of tackling this problem and demonstrates that a combination of simple heuristics with conventional sequencing approaches is unlikely to be effective in finding near-optimum feedorder/topology combinations." [49, p. 2712]

Simulated annealing provides another adaptive approach to the search for improved schedules. Where genetic algorithms derive from an analogy with population genetics, simulated annealing uses principles of statistical mechanics. Good introductions to simulated annealing are in [50, 51].

Musser, et al. [52] describe a simulated annealing approach to scheduling currently in use at a Texas Instruments PC-board manufacturing facility. The system is named the Annealing Based Experiment in Scheduling (ABES), and produces a weekly schedule for a process containing about 50 different operations. The authors report "favorable" reactions from personnel at the plant. However, the average time to generate a schedule is about 15 minutes, which precludes the system from functioning in a real-time, re-scheduling mode.

Van Laarhoven et al. [53] conducted a series of experiments using simulated annealing algorithms on a series of job shop scheduling problems ranging in size from six machines/six jobs to ten machines/thirty jobs. The authors conclude,

> ...a comparison of simulated annealing and tailored heuristics usually leads to the conclusion that tailored algorithms are more efficient and more effective than simulated annealing...Simulated annealing has the potential of (better solutions)...at the cost of large running times. [p. 124]

In an effort to improve the running time required by simulated annealing to a job shop problem, Ogbu and Smith [54] tried starting the annealing algorithm with a

good initial solution. The authors report "no discernible advantage" is obtained over a random starting solution. Also, [54] reports the choice of the perturbation scheme "affects the quality" of the solution.

Abramson [55] discusses a special architecture for scheduling problems using simulated annealing. Also presented is a procedure to measure the quality of a solution as a function of the seriousness of broken constraints. This procedure and the architecture can dramatically improve the run time of the simulated annealing approach to scheduling. Abramson [56] also applies a parallel annealing algorithm to construct school timetables.

In another attempt to speed-up the annealing algorithm, Mittenthal et al. [57] present a hybrid simulated annealing approach for the single machine scheduling problem. This procedure considers non-regular penalty functions, such as a situation where a penalty is imposed for tardy and early jobs. The hybrid approach "was to consider all feasible k-move and k-swap neighbors of a given V-shaped sequence and implement the change yielding the greatest decrease." [p. 106] The authors report near optimal solutions using this procedure with reasonable computing times.

In a typical job shop operation, machines are grouped by function. In a cellular manufacturing system, machines are dedicated to processing a specific number of "part families" [58]. Vakharia, and Chang [58] employ a heuristic based on simulated annealing to schedule part families in a manufacturing cell. (A thorough analysis of cellular manufacturing can be found in [59]). The primary difference between traditional job shop scheduling and cellular manufacturing is the effect of cell set-up times for individual part families. Therefore, the cellular manufacturing system is usually viewed as a two-part scheduling problem: one stage entails specifying a sequence of *all* part families, and the second stage concerns scheduling the jobs within *each* part family. The authors provide a detailed description of the simulated annealing algorithm for the cellular manufacturing process, and compare their results with several other procedures on a series of randomly generated data sets. Results indicate the simulated annealing algorithm provides better solutions than heuristics described in [60, 61] at a rather modest increase in computational time. For small problems, the simulated annealing procedure provided comparable solutions to integer programming models with considerably less computational time. For larger problems, the simulated annealing procedure provides better answers in less time than integer programming methods.

Brusco, et al. [62] employ a simulated annealing approach to the discontinuous labor scheduling problem, that is, a labor tour scheduling problem where the planning horizon is less than 24 hours each day. (A thorough review of the labor tour scheduling problem may be found in [63]). In this problem formulation, the objective is to minimize the number of full-time employees. Additionally, the

authors tackle a highly flexible problem considering items such as meal breaks and consecutive days off. A detailed description of the annealing algorithm is presented, including remarks on the computational effort and results. The authors state the algorithm was considerably faster than a standard branch and bound procedure and exceeded the answers found by the branch and bound algorithm by an average of only 1.36%.

While genetic algorithms and simulated annealing are CI approaches analogous to natural processes, neural networks attempt to combine processes analogies with structural analogies. In particular, these systems attempt to characterize the distributed parallel architecture and processing of networks of biological neurons. There are a wide variety of architectures and processing strategies that comprise the field of neural networks. Good introductions to the variety of approaches can be found in [64, 65].

Given the promise of neural networks for pattern recognition and search, their use in scheduling has become more prominent. Zhou, et al. [66] describe the use of a Hopefield network for job shop scheduling. They employ a linear cost function in place of the usual quadratic cost function to keep network complexity linear in the number of operations. The authors report very good results are obtained using their model.

One distinct problem encountered with Hopefield networks is the possibility of the network becoming "stuck" at some sub-optimal, possibly even non-feasible solution. Vaithyanthan and Ignizio [67] review some stochastic procedures, and present a new procedure applied to scheduling to force Hopefield networks out of local minimum. However, the authors point out there is no way to ensure local minima are not re-visited, thus the time needed to solve a problem using this approach could be exponential.

Satake, et al. [68] also use Hopefield networks in an attempt to minimize the makespan of the job-shop scheduling problem. This approaches differs from other methods in that threshold values of neurons are changed at each transition. Since this modification may lead to sub-optimal solutions, the authors employ a temperature mechanism, similar to that used in a Boltzman machine, in order to obtain better solutions. The temperature is lowered until no improvement is seen in the objective function for a specified number of iterations. The authors report reasonable computing time, and solutions that are optimal or close to optimal.

As with AI techniques, one promising direction is to combine methods from different areas to produce hybrid scheduling systems. CI methods have also successfully combined with each other and with techniques from OR to enhance the search for good solutions.

Liao [69] applies neural networks to the cellular manufacturing problem in a three-phase procedure aimed at reducing operating and material handling costs. In stage one, different part routings are evaluated, and the routing which minimizes operating costs is selected. A neural network is employed in stage two to determine how to group machines into designated cells. Finally, in stage three, STORM software is employed to produce a layout for the each cell. An example of the methodology is provided.

In an interesting combination of mathematical programming and CI techniques, Aoki, et al. [70] employ a backpropagation neural network to estimate the Lagrange multipliers for an integer 0-1 programming problem. The specific situation concerns scheduling generators in an electric power generation scheduling problem. The objective is to minimize cost, where the 0-1 variables represent the start-up/shut-down of generators, and continuous variables representing power outputs. Constraints include a balance between supply and demand, generator capacity, and operating constraints. A Lagrangian relaxation method is one way to solve this problem. As the authors point-out, the efficiency of the Lagrangian relaxation is highly dependent on how the Lagrange multipliers are calculated. The authors train a backpropagation neural network from data prepared form past records. Input to the network consists of power demand vectors, and the output is the Lagrange multipliers. Numerical results indicate a properly trained network is capable of predicting Lagrange multipliers which can be used to solve the power generation scheduling problem.

Willems and Rooda [71] translate a job shop scheduling problem into an integer linear program, and then apply neural networks to eliminate the need for integer adjustments. The authors first apply some elementary precalcualtions to determine the earliest possible start times of the operations. This precalculation process reduces the search space and speeds-up the solution process. Applications of this methodology in a simulated job shop environment shows the network provides feasible solutions without the need for integer adjustments.

Chryssolouris, et al. [72] use neural networks, in conjunction with simulation modeling, to assist in the design of manufacturing systems. This system does not directly schedule production, rather, it determines the job shop resources necessary for optimal performance. This system is a precursor to a scheduling problem, however, the methodology is general enough that it could be applied to scheduling problems.

Liang, et al. [73] integrate stochastic modeling with neural networks for real-time manufacturing scheduling. The system consists of three steps. The first step involves collecting data to train the neural network. The second step applies a semi-Markov decision model to remove inconsistent data from the training data. The third step uses the purified data in the neural network. The second step helps to filter out inconsistent data. Their experiments showed that

this approach was better than other approaches and that the difference was statistically significant.

Cihan and Sittisathanchai [74] combine genetic algorithms and neural networks into a hybrid genetic neuro-scheduler for the job shop scheduling problem. The authors report the systems always found the optimal solution for a 3 job, 4 machine problem. For a 10 job, 15 machine job shop, the system consistently outperformed a shortest processing time heuristic.

Finally, Toure, et al. [75] use a simulation routine, the dispatching rules described earlier, and neural networks to create a hybrid system for production scheduling that combines KBS, simulation, and neural networks. The authors selected six dispatching rules and created simulations that emphasize the strengths of each dispatching rule. Then, using a training set of 200 examples, the authors trained a backpropagation neural network to recognize when a particular dispatching rule is appropriate. The authors tested the network with 100 new cases, and reported the network is "consistent" in selecting the appropriate dispatching rule.

3. Transportation Scheduling

Transportation scheduling involves the allocation of conveyances and other resources for the movement of goods and people. The other resources can include labor, fuel, and maintenance assets. Again our goal is typically to minimize cost subject to constraints such as demand, conveyance type, conveyance capacity, and conveyance availability. Spatial concerns determine the routing of the conveyances and, thus also effectively constrain the choice of schedules.

Transportation scheduling problems are mathematically similar to production scheduling problems. Replacing the set of jobs with a set of locations, and considering the set of machines as the set of conveyance, then the problem is to assign locations to conveyances over time. A route through locations is similar to a process plan for product.

However, while production and transportation scheduling problems appear similar mathematically, their practical differences have necessitated somewhat different solution approaches. Transportation problems usually use the same conveyance for most movement (e.g. trucks), while it is rare for a product to remain on the same machine for all jobs. Transportation problems also tend to be much larger than production problems because the number of locations can be quite large. A bus routing problem in a moderately sized city can have over 10,000 potential stops.

In transportation a conveyance can move multiple items with different path objectives at the same time. This would be equivalent to a single machine in manufacturing making multiple different products at the same time. While this can happen, it is relatively unusual, while the problem is typical in transportation. Further, the transport objects are typically divisible (e.g. a train can haul multiple containers with different goods to different locations). In contrast machines in manufacturing typically work on one indivisible product.

Transportation problems also view rescheduling differently. Most information such as the number of available vehicles, the nature and location of demands, and the distance between cities, are known before the schedule is created. If one vehicle in a fleet of vehicles breaks down during movement, it is not usually necessary to recall the entire fleet and reschedule all the vehicles. However, if one machine in a production schedule breaks down, a bottleneck can occur if an quick rescheduling decision is not made.

Finally, unlike production scheduling, transportation problems are intimately linked to routing. Routing defines the paths of the conveyances in the transportation network. In production, the path of the product is normally defined by its design and process plan. In transportation the path has greater flexibility and in many cases the routing and scheduling problems are solved together. Of course the most famous routing problem – the Traveling Salesman Problem [76] – often lies at the heart of many transportation routing and scheduling problems.

As with production scheduling we can define three dimensions to the transportation scheduling problem:

1. Source of Requirements – requirements can come from sources external to the distribution organization (e.g. demand at the individual locations) or from organization itself (e.g. warehouse operations).
2. Process Complexity – this involves the number, type, and characteristics of the conveyances, locations, and items to be moved.
3. Selection Criteria – describes the evaluation procedures for choosing among schedules. Examples include: average travel time, total travel cost, total distance, deadhead time, fuel cost, maintenance cost, and the number of required conveyances. As with production scheduling objectives can conflict with each other requiring the use of multi-objective methods.

As with production scheduling the OR community has produced an impressive array of exact and inexact methods for transportation scheduling. A survey of these methods is in [77, 78]. Good examples of exact methods, which employ

cutting-plane or branch and bound algorithms are in Baker [79] and Dell'Amico et al. [80].

The complexity of most real transportation routing and scheduling problems has lead to the development of a sizable number of inexact methods. For example, relaxation methods use mathematical programming algorithms to solve less constrained versions of the original problem (see Stewart and Golden [81] and Bertossi, et al. [82]).

Another popular inexact approach starts with a feasible solution and performs interchanges between routes to improve the value of the objective function (see Christofidies and Eilon [83], Waters [84], and Fahrion and Wrede [85]). Unlike relaxation methods, this approach starts with a feasible solution and maintains feasibility throughout the search. A heuristic related to the interchange heuristic starts with a (possibly infeasible) solution and compares the current configuration with an alternate configuration. The algorithm inserts nodes which are the least expensive according to some criteria (see Clarke and Wright [86] and Ahn and Shin [87]).

Yet another inexact procedure employs partitioning of the nodes or arcs in the transportation network. For example, the nodes or arcs can be partitioned based on proximity and then routes are found over each element of the partition (see Wren and Holliday [88] and Gillet and Miller [89]). Alternatively, we can start with one large route and look for suitable (e.g. feasible and low cost) partitions of it (see Newton and Thomas [90] and Beasley [91]).

While exact and inexact methods attempt to find the route or schedule automatically, others have suggested interactive methods that include human-in-the-loop control of the search process (See Krolak, et al. [92] and Cullen, et al. [93]). These methods also have the advantage of allowing the user to interactively address trade-offs between competing objectives.

Although these approaches have provided some successes, there remain a vast number of apparently intractable transportation scheduling problems. These have become the concern for methods employing AI or CI as described in the next two subsections.

3.1 AI Methods for Transportation Scheduling

As we noted with production scheduling, most of the AI work in transportation scheduling uses knowledge-based systems (KBSs). Again the emphasis is on rules for knowledge representation. However, unlike production scheduling general dispatching rules do not seem to exist in transportation. In fact, Waters [94], argues that generalized procedures will not have the flexibility needed for real problems.

Instead, Waters highlights the premium placed on the human transportation scheduler's ability to recognize patterns and take into account the subjective nature of many of the objectives and constraints. This point of view is strengthened by numerous case studies of real transportation scheduling problems. For example, Brown and Graves [95] state "Fundamental to the philosophy of the system (of dispatching petroleum trucks) is that the human dispatcher cannot be replaced." [p. 20]

Waters [84, 94] provides a general framework for using expert systems in relation to vehicle routing and scheduling problems. The approach is interactive, and centered on building petal-shaped routes and iteratively improving these routes. This implies taking larger problems and dividing them into smaller, more easily solved problems. This technique is also known as the divide and conquer technique. Note the relationship between this approach advocated for KBSs and the same divide and conquer strategy employed through various heuristics in the inexact methods discussed in the previous subsection.

An example of a successful KBS for transportation scheduling is the National Dispatch Router (NDR) [96]. NDR employs heuristic functions encoded in LISP to schedule a fleet of trucks at the Digital Equipment Corporation (DEC) without reducing customer service. NDR is estimated to save DEC about $1 million annually.

Another working KBS does not actually schedule conveyances. However, it does schedule and assign gates to aircraft. This system is the Gate Assignment Display System (GADS) [97], employed by United Airlines. The system works to reduce travel delays at Chicago's O'Hare and Denver's Stapleton airports. GADS communicates with United's UNIMATIC flight information system which tracks all United flights. Prior to the creation of GADS, gate assignments were done manually. The manual system used a wall-sized scheduling board and relied on institutional memory for details such as which gates could accommodate which type aircraft.

Pang [98] describes an expert system that acts in real-time to control and schedule elevators. The system makes explicit decisions to schedule a group of elevators quickly and efficiently. A noteworthy aspect of Pang's system is the use of a blackboard architecture to schedule the elevators. A blackboard controls cooperating knowledge sources. The blackboard architecture allows the system to operate in real-time.

An unusual approach to the vehicle routing problem concentrates on assisting the algorithm designer rather than the transportation scheduler. Potvin, et al. [99] describe an interactive computer program that helps an algorithm designer build a better and more efficient routing algorithm The system, named ALTO

(Algorithmes de tournees (routing algorithms)), uses general templates which employ generic routing problem solving procedures. The motivation for ALTO stems from the belief that researchers "create a first draft of an algorithm by closely analyzing the characteristics of the problems to be solved, apply the algorithm and, if the results are unsatisfactory, iteratively refine the algorithm until 'good' solutions are obtained." [p. 517]. Currently, ALTO is limited to single depot node routing problems in which a request for goods is located on specific nodes of a transportation network.

3.2 CI Methods for Transportation Scheduling

As with production scheduling, genetic algorithms, simulated annealing, and neural networks are the primary CI methods that have been used for transportation scheduling. However their use in transportation scheduling has been more limited.

An evolution program named GENOCOP [100, 101] provides a domain independent methodology for handling linear constraints. The main ideas behind GENOCOP are to eliminate constraints by replacing variables with linear combinations of other variables and to design special genetic operators which maintain the integrity of the constraints. Results from GENOCOP were not as good as those obtained using linear programming techniques on a linear transportation problem [101]. However, GENOCOP usually outperforms GAMS [102], a program using standard nonlinear programming techniques, on transportation problems with nonlinear objective functions.

A second evolution program, named GENETIC-2 [100, 103, 104], uses a nonstandard genetic algorithm to solve a balanced (total supply equals total demand) nonlinear transportation problem. The initialization process starts with a feasible solution with a process similar to that used in the Simplex Method for linear programming [p. 310]. Again, results indicate that GAMS performs better on smooth, monotonic functions. However, GENETIC-2 outperforms GAMS on other functions, such as step functions.

Brown and Huntley [105] used simulated annealing to effectively route and schedule grain trains. Their approach employed a very efficient representation that combined the routing and scheduling problems for solution by a single search mechanism. Their system also provided an easily understood and usable interface, which enhanced its acceptance.

Wright [106] compares deterministic, local improvement, and simulated annealing algorithms for the problem of scheduling locomotives to cover all trains for a given timetable. The author reports, "(the) Annealing algorithm appears to perform better than simple local improvement algorithms from random starting solutions." [p. 192]

Alfa et al. [107] combine simulated annealing with the 3-opt procedure to solve some large vehicle routing problems. The application is straight-forward, and the authors report obtaining results as good as the best known solution on two of three problems tested.

Potvin, et al. [108] describe the use of backpropagation neural networks for dispatching vehicles to customers requesting immediate service. The authors compared the neural network solutions with those of an expert dispatcher on 100 cases. They concluded that a neural network approach can reliably determine the assignment of a new customer to a vehicle. Additionally, the neural network can implicitly model a host of objectives, such as minimizing customer waiting time and minimizing travel distance. Even when the neural network solution differed from the expert, the solutions provided were always acceptable.

A hybrid of OR and neural networks for vehicle routing was developed by Nygard, et al. [109]. The specific problem they examined was to route a fleet of vehicles with known capacity through a set of delivery points, starting and ending at a single depot. The authors trained a neural network to recognize when to apply an appropriate algorithm to various characterizations of the problem. The authors used a training set of four different algorithms and five different problem categories. Each algorithm was used to solve each problem. Accuracy of the algorithms was measured based on the objective of minimizing mileage, thus, the best algorithm for a particular routing problem was the one with the total shortest distance. After the network was trained, the authors presented the network with 100 test problems. The neural network selected the algorithm that minimized the distance 83 times. When the network was wrong, it never selected the worst algorithm and always selected an algorithm where the difference in distance was quite small.

Lin et al. [110] combine genetic algorithms and simulated annealing to form a hybrid approach in solving some NP-Hard problems like the traveling salesman or multiconstraint knapsack problem. This methodology uses genetic operators to create a population of solutions at each temperature in order to "induce maturation." Rules are imposed to ensure the value of offspring are less than the average cost of the preceding generation. The authors state this rule ensures convergence of the algorithm while shortening the length of the Markov Chain at each generation.

4. Project Scheduling

Project scheduling allocates resources at specified times to tasks which together complete a project or accomplish a mission. While production or transportation scheduling involve recurrent operational decisions, project scheduling normally

involves a unique project. The project may be the construction of a house or a deep space expedition by an unmanned probe. The principles involved in project scheduling carry over from one project to another, but we are never scheduling exactly the same objects for the same reasons as we do in production and transportation.

As with the previous two application areas we can view project scheduling along three dimensions:

1. Source of Requirements – requirements in project scheduling are defined strategically by choice of activities and tactically by the resources used to complete the activities. These choices highlight a major difference with the previous two application areas. In production we have a set of well-defined jobs that we must complete to successfully manufacture the product. Similarly in transportation scheduling we know the characteristics of the items to be transported and the capabilities of the conveyances. By contrast in project scheduling we have flexibility in the choice of activities to complete the project and in the mechanism or resources to perform the activities. For example, in constructing a house we may choose any of a number of methods and materials to waterproof the foundation. This choice can dramatically change the completion time as can the number and experience of the workers assigned to the task.
2. Process Complexity – this concerns the number, type, and characteristics of the activities and their precedence relationships. Most project scheduling problems are represented by a network diagram which delineates the activities and their precedence relationships.
3. Selection Criteria – describes the evaluation procedures for choosing among schedules. Examples include: completion time, total project cost, project quality, idle time, probability of schedule overrun. As with the previous applications production scheduling objectives can conflict with each other. Most notable in this regard are completion time and total project cost, typically called cost and schedule.

Approaches to project scheduling have a rich history dating back more than 30 years ago to address project management issues in large defense programs. Two early methods, in particular, became almost universally associated with scheduling systems: the Critical Path Method (CPM) and the Project Evaluation and Review Technique (PERT). Because of this early start there exist a number of successful commercial packages to assist in project scheduling (see [111]).

CPM and PERT viewed the project scheduling problem from somewhat different perspectives. CPM considers activity times as a function of cost and seeks to trade-off cost with completion time. PERT models activity times as random variables and is used to analyze project completion time. Here we briefly consider some of the successors to these techniques which employ exact and inexact algorithms. More in-depth reviews of these PERT and CPM approaches are in [112, 113, 114, 115].

Exact algorithms have found a place in scheduling project activities for well-defined problem formulations with limited complexity. The simplest version of CPM seeks the earliest completion time as the maximum length path through the activity network. A number of algorithms can solve this problem exactly – simplex, network flow, and Dijkstra's. These same algorithms can be employed for extended CPM techniques which allow more detailed precedence relationships (see the Precedence Diagramming Method [113]).

CPM was designed to model time-cost relationships at each activity. The original approach assumed a linear relationship and was solved using linear programming. When the time-cost relationship is nonlinear and convex then Hendrickson and Janson [116] use nonlinear programming to obtain exact solutions. For concave but monotonic time-cost relationships Falk et al. [117] employ branch-and-bound. Finally, more general time-cost relationships can be solved exactly with dynamic programming as shown by Elmaghraby [112].

Exact algorithms have also been used with resource constrained problems. Here, the resources available for the activities have constraints beyond normal and crash times. Patterson [118] looked as several exact algorithms and concluded that branch-and-bound proposed by Stinson et al. [119] was the most effective. Willis and Hastings [120] have successfully employed branch-and-bound, while Petrovic [121] uses dynamic programming.

Finally, exact methods have been used for combining the choice of activities with the choice of schedule (see Decision CPM [122]). While the original authors of this approach used an inexact solution algorithm, Hindelang and Muth [123] later developed an exact dynamic programming approach.

Exact algorithms have the same shortcoming for project scheduling that they had for production and transportation scheduling – they can only solve relatively small problems. For this reason a wide variety of inexact methods or heuristics have been advocated. Davis and Patterson [124] review eight different heuristics for the resource constrained problem. Kurtulus and Davis [125] also examined different heuristic approaches and showed that combinations of heuristics performed best. This result suggests that intelligent systems that can adapt to the specific characteristics of the problem domain have potential to further enhance project scheduling methods.

4.1 AI Methods for Project Scheduling

Zong, et al. [126] describe an expert system employed in project scheduling that minimizes the duration of the project. This expert system applies an exchange heuristic that takes an initial schedule and improves upon it by rearranging operations.

Zozaya-Gorostiza, et al. [127] developed PLANEX to perform process planning in construction and manufacturing. The system begins with an initial description of the goal of the project (i.e. the artifact to be built). From this PLANEX creates and links the activities needed to complete the project. The system also assigns resources. PLANEX can be used across a variety of domains by adding to its knowledge sources.

Barber, et al. [128] describe a prototype for an intelligent tool for project control and scheduling known as XPERT (expert project evaluation reasoning tool kit). XPERT considers the problem of expediting selected activities if a project is to meet some type of timing constraints. For example, if a project experiences work slippage, the effects of penalty clauses and overtime pay might be considered in deciding which activities to expedite.

In Barber et al. [129], the authors evaluate the XPERT system using three case studies and the expertise of senior project managers at a major contracting firm. Results indicate, "The prototype systems proved that the ideas were sound, but that more work is required before such systems can be realistically delivered." [p. 50] The authors also provide interesting comments regarding the interaction of knowledge engineers and domain experts, for example, the experts state, "...preventing projects from missing deadlines is a far better approach than trying to recover projects which have already slipped." [p. 36] This type of heuristic information is exactly the type of rules which justify the use of a knowledge-based approach to project scheduling.

4.2 CI Methods for Project Scheduling

Lawton [130] presents an introductory lesson in the application of genetic algorithms to project scheduling. A simple algorithm is described, highlighting the basic procedures common to all genetic algorithms. The actual implementation of the algorithm and specific results are not discussed.

Hou, et al. [131] use genetic algorithms to solve a multiprocessor scheduling problem. The multiprocessor scheduling problem is defined as, "scheduling a set of partially ordered computational tasks onto a multiprocessor system so that a performance criteria will be optimized." [p. 113] The problem is depicted as a "task graph" and is analogous to a project scheduling problem. The authors

provide a complete description of the string representation, initial population generation, and genetic operators used in this problem formulation. Results show the genetic algorithm generally provides solutions close to optimality in reasonable computing times.

McKim [132] uses three case studies to compare neural networks against traditional project management techniques. Results indicate, depending on the sophistication of the network, the predicative capabilities of the neural network resulted in a 5.8% to 94% cost savings.

Johnston and Adorf [133] create long-term schedules for the Hubble Space Telescope using neural networks. The authors employ a feedforward network, with a stochastic, sequential updating rule called a Guarded Discrete Stochastic (GDS) network [134]. This type of network requires no training, instead the "network is designed entirely and automatically from the scheduling constraints, both strict and preference." [p. 210]. Additionally, [134] provides an excellent bibliography concerning scheduling and neural networks.

Kumar et al. [135] compare the theories of fuzzy sets and neural networks for the purpose of software project planning and control. The authors compare 64 fuzzy associative memory (FAM) rules to a feedforward neural network trained with the same 64 FAM rules. Results highlight the general differences between rule-based and neural network approaches. For example, the authors remark that the FAM rules are distinct, allowing the user to determine exactly how a rule contributed to a given answer. However, in a neural network, node contributions are usually not discernible.

5. Conclusions

This paper has surveyed a wide range of approaches to scheduling. These "intelligent systems" approaches have been inspired by fields of study ranging from statistical mechanics, genetics, and the biological underpinnings of the human brain to traditional statistics and function optimization. At first glance, such a variety might indicate that scheduling lacks a central focus or strong foundations. However, Simon and Newell [136] called for just such amalgamation of techniques in their landmark speech to the Operations Research Society of America in 1957. They stated,

> In some ways it is a very new idea to draw upon the techniques and fundamental knowledge of these fields (mathematics, physical scientists, biologists, statisticians, economists, and political scientists) in order to improve the everyday operation of administrative organizations. [p. 1]

Thus, the idea of researchers from diverse fields pooling their talents to solve scheduling problems is really an indication of the strength and vitality of the field.

Before looking to the future for potential areas of research, we think it is important to look to the past and examine some of the common mistakes found in systems designed to solve scheduling problems. Below is a summary of the most common problem areas in production scheduling systems as outlined by a recent panel discussion [137]:

1. Inadequate understanding of the dominant problem characteristics, for example, what causes bottlenecks in a particular system.
2. Reliance on locally greedy strategies without taking into account the synergistic effect of local decisions.
3. Reliance on the shallow expertise of the human scheduler without fully understanding the actual process involved.
4. Excess concern about trivialities. For example, a proposed eight-hour schedule might have two machines overlapping for one minute, seven hours from the present time. This overlap is meaningless at the beginning of the production schedule.
5. Solving a too simplified version of the problem in the prototype stage. This can lead to a system that is incapable of solving the actual problem at-hand.

Analysis of the above list, coupled with an understanding of the solution techniques presented in this paper, points to several potential areas of future research. AI-based approaches offer the well-known advantages of symbolic systems: they are easy to understand and they perform like human experts. KBS's, sometimes called the practical side of AI, are currently saving companies millions of dollars a year by providing better scheduling solutions, not optimal ones. However, these KBS's suffer from long and tedious development efforts and sometimes include human errors as part of the solution. Additionally, KBS's often employ simple scheduling rules and are therefore hindered by the inability to generate quality solutions in a reasonable time when used to solve large-scale scheduling problems. A major area for research in AI methods involves the use of machine learning. The addition of these techniques will speed for development and the adaptability of the resulting systems to changing operational conditions.

There is clearly a need to better understand the potential of CI techniques, like neural networks, for performing scheduling operations. Our survey here revealed a particular shortage of CI work in project scheduling. Neural networks, simulated annealing, and genetic algorithms also seem to offer excellent potential for adapting scheduling algorithms to changing conditions.

This seems to be important in all three of the domains we surveyed. As yet, the potential is unfulfilled by existing systems.

We envision that future scheduling systems will be intelligent hybrid systems that are hierarchical in nature, with the higher user interface levels employing knowledge-based concepts and lower-level modules based-on computational intelligence, inexact algorithms, and mathematical programming. Unfortunately, given the complex and diverse nature of scheduling problems, finding a unifying theory for this large class of problems may prove impossible. Improvements in the enabling technologies, such as AI, CI, and OR, will allow for more complex problems to be solved but are likely to be incremental in nature. Advances in integrated or hybrid systems will offer the most radical improvements for solving classes of practical problems.

Scheduling is referred to as an intractable problem, however, everyday human schedulers provide solutions for these unyielding problems. Therefore, it is not a question of whether scheduling problems are solvable. It is a question of solution quality and timeliness. These intelligent approaches to production and transportation scheduling problems represent some of the latest technology applied to one of mankind's earliest logistical problems.

BIBLIOGRAPHY

1. Simon, Herbert, A., "Two Heads Are better than One: The Collaboration between AI and OR", *Interfaces,* Vol. 17, No. 4, 1987, pp. 8-15.

2. Spitzer, Murry, "The Computer Art of Schedule-Making", *Datamation,* Apr. 1969, pp. 84-86.

3. McKay, Kenneth N., Frank R. Safayeni, and John A. Buzacott, "Job-Shop Scheduling Theory: What is Relevant?" *Interfaces,* Vol. 18, No. 4, 1988, pp. 83-90.

4. Sussams, J., "The Future for Computerized Vehicle-Load Planning Systems", *J. of Opl. Res. Soc.,* Vol. 35, 1982, pp. 963-966.

5. Garey, Michael R., and David S. Johnson. *Computers and Intractability.* W.H. Freeman, 1979.

6. Lenstra, J.K., and A.H. G. Rinnooy Kan, "Complexity of Vehicle Routing and Scheduling Problems", *Networks,* Vol. 11, 1981, pp. 221-227.

7. Newell, A., and H.A. Simon, "Computer Science as Empirical Inquiry: Symbols and Search", *Comm. of the ACM*, Vol. 19, No. 3, 1976, pp. 113-126.

8. Waterman, Donald A. *A Guide to Expert Systems.* Addison-Wesley, 1986.

9. Holland, John H. *Adaptation in Natural and Artificial Systems.* University of Michigan Press, 1975.

10. Fogel, L.J., A.J. Owens, and M.J. Walsh, *Artificial Intelligence Through Simulated Evolution.* Wiley, 1966.

11. Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing.", *IBM Computer Science/Engineering Technology Report,* IBM Thomas J. Watson Research Center, 1982.

12. Kirkpatrick, S., C. D. Gelatt, Jr., and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, Vol. 220, 1983, pp. 671-680.

13. Farlow, S.J. (ed.) *Self Organizing Methods in Modeling: GMDH Type Algorithms.* Marcel Dekker Inc., 1984.

14. Nilsson, N.J., *Principles of Artificial Intelligence*, Morgan Kaufmann, 1980.

15. Graves, Stephen C., "A Review of Production Scheduling", *Ops. Res.*, Vol. 29, No. 4, 1981, pp. 646-675.

16. Hadavi, Khosrow, Wan-Ling Hsu, Tony Chen, and Cheoung-Nam Lee, "An Architecture for Real-Time Distributed Scheduling", *AI Mag.*, Vol. 13, Fall 1992, pp. 46-56.

17. Baker, Kenneth R., *Introduction to Sequencing and Scheduling.* Wiley, 1974.

18. French, Simon, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop* Ellis Horwood Limited, 1982.

19. Rodammer, F.A., and K.P. White, Jr., "A Recent Survey of Production Scheduling", *IEEE Trans. on Sys., Man, and Cybernetics*, Vol. 18, No. 6, 1988, pp. 841-852.

20. White, K.P., Jr., "Advances in the Theory and Practice of Production Scheduling", in C.T. Leondes (ed.), *Advances in Control and Dynamic Systems*, Academic Press, 1990.

21. Ow, Peng Si, Stephen F. Smith, and Randy Howie, "A Cooperative Scheduling System" in Michael D. Oliff , (ed.) *Expert Systems and Intelligent Manufacturing,* North-Holland, 1988, pp. 43-56.

22. Glover, Fred, "TABU Search Part I", *ORSA J. on Comput.*, Vol. 1, No. 3, 1989, pp. 190-206.

23. Glover, Fred, "TABU Search Part II", *ORSA J. on Comput.*, Vol. 2, No. 1, 1990, pp. 4-32.

24. Glover, Fred, "Heuristics for Integer Programming using Surrogate Constraints", *Dec. Sci.*, Vol. 18, No. 1, 1977, pp. 156-166.

25. Widmer, M., "Job Shop Scheduling with Tooling Constraints: a Tabu Search Approach", *J. of Opl. Res. Soc.*, Vol. 42, No. 1, 1991, pp. 75-82.

26. Glover, Fred., "Tabu Search for Nonlinear and Parametric Optimization (with Links to Genetic Algorithms)", *Discrete Applied Math.* Vol. 49, 1994, pp. 231-255.

27. Ignizio, J.P., *Introduction to Expert Systems.* Mcgraw-Hill, 1991.

28. Blackstone, John H., Don T. Phillips, and Gary L. Hogg, "A State-of-the-Art Survey of Dispatching Rules for Manufacturing Job Shop Operations", *Int. J. Prod. Res.,* Vol. 20, No. 1, 1982, pp. 27-45.

29. Conway, R. W., and W. L. Maxwell, "Network Dispatching by Shortest Operation Discipline", *Ops. Res.*, Vol. 10, 1962, p. 51.

30. Elvers , D. A., "Job Shop Dispatching Rules Using Various Delivery Date Setting Criteria", *Prod. Inventory Mgmt.,* Vol. 14, 1973, p. 61.

31. Conway, R. W., "Priority Dispatching and Job Lateness in a Job Shop", *J. Ind. Eng.,* Vol. 16, 1965, p. 123.

32. Rochette, R., and R.P. Sadowski, "A Statistical Comparison of the Performance of Simple Dispatching Rules for a Particular Set of Job Shops", *Int. J. of Prod. Res.*, Vol. 14, 1976, p. 63.

33. Grant, T.J., "Lessons for O.R. from A.I.: A Scheduling Case Study", *Ops. Res.,* Vol. 37, No. 1, 1986, pp. 41-57

34. Pierreval, Henri, and Henri Ralambondrainy, "A Simulation and Learning Technique for Generating Knowledge about Manufacturing Systems Behavior", *J. of Opl. Res. Soc.,* Vol. 41, No. 6, 1990, pp. 461-474.

35. Iron Age, "Gulf States Steel Stays on Schedule", *Iron Age,* Vol. 8, Aug. 1992, pp. 25-26.

36. Fox, Mark S., and Stephen F. Smith, "ISIS—A Knowledge-based System for Factory Scheduling,", *Expert Systems,* Vol. 1, No. 1, 1984, pp. 25-49.

37. Ow, Peng SI, and Stephen F. Smith, "Two Design Principles for Knowledge-Based Systems", *Dec. Sci.,* Vol. 18, Summer 1987, pp. 430-447.

38. Elleby P., H. E. Fargher, and T. R. Addis, "Reactive Constraint-Based Job-Scheduling" in Michael D. Oliff, (ed.) *Expert Systems and Intelligent Manufacturing,* North-Holland, 1988, pp. 1-10.

39. Bensana, E., G. Bel, and D. Dubois, "OPAL: A Multi-Knowledge-Based System for Industrial Job-Shop Scheduling", *Int. J. Prod. Res.,* Vol. 26, No. 5, 1988, pp. 795-819.

40 Kusiak, Andrew, "A Knowledge and Optimization-Based Approach to Scheduling in Automated Manufacturing Systems", in Donald E Brown, and Chelsea C. White, III, (eds) *Operations Research and Artificial Intelligence.* Kluwer Academic Publishers, 1990, pp. 453-479.

41. Binbasioglu, M. and M. Jarke, "Knowledge-Based Formulation of Linear Planning Models", in H.G. Sol, C.A.T. Takkenberg and P.F. DeVries Robbe (eds.) *Expert Systems and Artificial Intelligence in Decision Support Systems*, 1987.

42 Davis, Lawrence, "A Genetic Algorithm Tutorial", in Lawrence Davis, (ed.), *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, 1991.

43. Goldberg, David E., *Genetic Algorithms in Search, Optimization, and Machine Learning,* Addison-Wesley, 1989.

44. Biegel, John E., and James J. Davern, "Genetic Algorithms and Job Shop Scheduling", *Computers & Ind. Eng.*, Vol. 19, No. 1-4, 1990, pp. 81-91.

45. Gupta, Mahesh, Yash Gupta, and Anup Kumar, "Minimizing Flow time Variance in a Single Machine System using Genetic Algorithms", *European J. Ops. Res.*, Vol. 70, No. 3, 1993, pp. 289-303.

46. Whitley, Darrell, Timothy Starkweather, and Daniel Shaner, "The Traveling Salesman and Sequence Scheduling: Quality Solutions using Genetic Edge Recombination", in Lawrence Davis, (ed), *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, 1991, pp. 350-371.

47. Syswerda, Gilbert, "Schedule Optimization Using Genetic Algorithms", in Lawrence Davis, (ed), *Handbook of Genetic Algorithms,* Van Nostrand Reinhold, 1991, pp. 332-349.

48. Uckun, Serdar, Sugato Bagchi, Kazuhiko Kawamura, and Yutaka Miyabe, "Managing Genetic Search in Job Shop Scheduling", *IEEE Expert,* Oct. 1993, pp. 15-24.

49. Cartwright, Hugh M., and Robert A, Long, "Simultaneous Optimization of Chemical Flowshop Sequencing and Topology Using Genetic Algorithms", *Ind. Eng. Chem. Res.*, Vol. 32, 1993, pp. 2706-2713.

50. Aarts, E. and J. Korst, *Simulated Annealing and Boltzmann Machines*, Wiley, 1989.

51. Aarts, E. and P.J.M. Var Larhoven, *Simulated Annealing: Theory and Applications*. Kluwer Academic, 1988.

52. Musser, K.L., J.S. Dhingra, and G.L. Blakenship, "Optimization Based Job Scheduling", *IEEE Trans. on Automatic Control,* Vol. 38, No. 5, May 1993, 808-813.

53. Van Laarhoven, Peter J., Emile H. Aarts, and Jan Karel Lenstra, "Job Shop Scheduling by Simulated Annealing", *Ops. Res.,* Vol. 40, No. 1, 1992, pp. 113-125.

54. Ogbu, F.A., and D.K. Smith, "The Application of the Simulated Annealing Algorithm to the Solution of the n/m/C Max Flowhop Problem", *Computers & Ops. Res.*, Vol. 17, No. 3, 1990, pp. 243-253.

55. Abramson, David, "A Very High Speed Architecture for Simulated Annealing", *Computer,* May 1992, pp. 27-36

56. Abramson, D, "Constructing School Timetables Using Simulated Annealing: Sequential and Parallel Algorithms", *Mgmt. Sci.*, Vol. 37, No. 1, 1991, pp. 98-113.

57. Mittenthal, John, Madabhushi Raghavachari, and Arif Rana, "A Hybrid Simulated Annealing Approach for Single Machine Scheduling Problems with Non-Regular Penalty Functions", *Computers Ops. Res.*, Vol. 20, No. 2, 1993, pp. 103-111.

58. Vakharia, Asoo J., and Yih-Long Chang, "A Simulated Annealing Approach to Scheduling a Manufacturing Cell", *Naval Res. Logistics*, Vol. 37, 1990, pp. 559-577.

59. Burbidge, J.L., *The Introduction of Group Technology*, Wiley, 1975.

60. Campbell, H.G., R.A. Dudek, and M.L. Smith, "A Heuristic Algorithm for the *n* Job, *m* Shop Sequencing Problem", *Mgmt. Sci.*, Vol. 16, No. 10, 1970, pp. 630-637.

61. Nawaz, M, E. Enscore, and I. Ham, "A Heuristic Algorithm for the *m*-Machine, *n*-Job Flow Shop Sequencing Problem", *OMEGA*, Vol. 11, No. 1, 1983, pp. 91-95.

62. Brusco, Michael J., and Larry W. Jacobs, "A Simulated Annealing Approach to the Solution of Flexible Labour Scheduling Problems", *J. of Opl. Res. Soc.,* Vol. 44, No. 12, 1993, 1191-1199.

63. Bechtold, S., M. Brusco, and M. Showater, "A Comparative Evaluation of Labor Tour Scheduling" *Dec. Sci.*, Vol. 23, pp. 683-699.

64. Wasserman, Philip D., *Neural Computing: Theory and Practice.* Van Nostrand Reinhold, 1989.

65. Hecht-Nielsen, Robert, *Neurocomputing.* Addison-Wesley, 1991.

66. Zhou, D.N., V. Cherkassky, T.R. Baldwin, and D.E. Olson, "A Neural Network Approach to Job-Shop Scheduling", *IEEE Trans. on Neural Networks,* Vol. 2, No. 1, Jan. 1991, pp. 175-179.

67. Vaithyanthan, Shivakumar, and James P. Ignizio, "A Stochastic Network for Resource Constrained Scheduling", *Computers Ops. Res.,* Vol. 19, No. 3/4, 1992, pp. 241-254.

68. Satake, T., K. Morikawa, and N. Nakamura, "Neural-Network Approach for Minimizing the Makespan of the General Job-Shop", *I. J. Prod. Econ.*, Vol. 33, 1994, pp. 67-74.

69. Liao, T., "Design of Line-Type Cellular Manufacturing Systems for Minimum Operating and Material-Handling Costs", *Int. J. Prod. Res.*, Vol. 32, No. 2, 1994, pp. 387-397.

70. Aoki, Kenichi, Masakazu Kanezashi, Masaru Itoh, and Haruki Matsuura, "Power Generation Scheduling by Neural Networks", *Int. J. Systems Sci.*, Vol. 23, No. 11, 1992, pp. 1977-1989.

71. Willems, T., and J. Rooda, "Neural Networks for Job-Shop Scheduling", *Control Eng. Practice*, Vol. 2, No. 1, 1994, pp. 31-39.

72. Chryssolouris, G., M. Lee, J. Pierce, and M. Domroese, "Use of Neural Networks for the Design of Manufacturing Systems", *Manufacturing Res.,* Vol. 3, No. 3, Sept. 1990, pp. 87-194.

73. Liang, Ting-Peng, Herbert Moskowitz, and Yuehwern Yih, "Integrating Neural Networks and Semi-Markov Processes for Automated Knowledge Acquisition: An Application to Real-time Scheduling", *Dec. Sci.,* Vol. 23, No. 6, 1992, pp. 1297-1314.

74. Dagli, Cihan and Sinchai Sittisathanchai, "Genetic Neuro-Scheduler for Job Shop Scheduling", *Computers and Ind. Eng.,* Vol. 25, No. 1-4, 1993, pp. 267-270.

75. Toure, Serge, Luis Rabelo, and Tomas Velasco, "Artificial Neural Networks for Flexible Manufacturing Systems Scheduling", *Computers and Ind. Eng.,* Vol. 25, No. 1-4, 1993, pp. 385-388.

76. Lawler, Eugene L. (ed.), *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, Wiley, 1990.

77. Bodin, L., B. Golden, A. Assad, and M. Ball, "Routing and Scheduling of Vehicles and Crews — The State of the Art", *Computers and Ops. Res.*, Vol. 10, No. 2, 1983, pp. 62-211.

78. Bodin, L., and B. Golden, "Classification in Vehicle Routing and Scheduling", *Networks*, Vol. 11, 1981, pp. 97-108.

79. Baker, Edward K., "An Exact Algorithm of the Time Constrained Traveling Salesman Problem", *Ops. Res.,* Vol. 31, 1993, pp. 938-945.

80. Dell'Amico, M., M. Fischetti, and P. Toth, "Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem", *Mgmt. Sci.,* , Vol. 39, No. 1, 1993, pp. 115-125.

81. Stewart, W., and B. Golden, "A Vehicle Routing Algorithm Based on Generalized Lagrange Multipliers", *Procc. of the AIDS 1979 Annual Conv, IL. More. E. Moore, B. Taylor (eds.), New Orleans, LA., 1979, pp. 108-110.*

82. Bertossi, A., P. Carraresi, and G. Gallo, "On Some Matching Problems Arising in Vehicle Scheduling Models", *Networks,* Vol. 17, 1987, pp. 271-281.

83. Christofides, N. and S. Eilon, "An Algorithm for the Vehicle-dispatching Problem", *Ops. Res. Qtrly.,* Vol. 2. No. 3, 1969, pp. 3309-318.

84. Waters, C.D.J., "A Solution Procedure for the Vehicle-Scheduling Problem Based on Iterative Route Improvement", *J. of Opl. Res. Soc.,* Vol. 38, No. 8, 1987, pp. 833-839.

85. Fahrion, R., and M. Wrede, "On a Principle of Chain-exchange for Vehicle-routeing Problems (1-VRP)", *J. of Opl. Res. Soc.,* Vol. 41, No. 9, 1990, pp. 821-827.

86. Clarke, G., and J. M. Wright, "Scheduling of Vehicles from a Central Depot to a Number of Delivery Points", *Ops. Res.,* Vol. 12, No. 4, 1964, pp. 568-580.

87. Ahn, B., and J. Shin, "Vehicle-routeing with Time Windows and Time-varying Congestion", *J. of Opl. Res. Soc.,* Vol. 42, No. 5, 1991, pp. 393-400.

88. Wren, A, and A. Holliday, "Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points", *Ops. Res. Qtrly.,* Vol. 23. No. 3, 1969, pp. 333-344.

89. Gillet, B., and L. Miller, "A Heuristic Algorithm for the Vehicle Dispatch Problem", *Ops. Res.,* Vol. 22, 1974, 340-349.

90. Newton, R., and W. Thomas, "Bus Routing in a Multi-School System", *Computers Ops. Res.,* Vol. 1, 1974, pp. 213-222.

91. Beasley, J.E., "Route First-Cluster Second Methods for Vehicle Routing", *OMEGA*, Vol. 11, No. 4, 1983, pp. 403-408.

92. Krolak, P., W. Felts, and G. Marble, "A Man-Machine Approach toward Solving the Traveling Salesman Problem", *Comm. ACM,* Vol. 14, 1971, pp. 327-334.

93. Cullen, F.H., J.J. Jarvis, and H.D. Ratliff, "Set Partitioning Based Heuristic for Interactive Routing", *Networks,* Vol. 11, 1981, pp. 125-144.

94. Waters, C. D. J, "Interactive Vehicle Routing", *J. of Opl. Res. Soc.*, Vol. 35, No. 9, 1984, pp. 821-826.

95. Brown, G.G., and G. W. Graves, "Real-Time Dispatch of Petroleum Tank Trucks", *Mgmt Sci,* Vol. 27, 1981, pp. 19-32.

96. Heller, Martin, "AI in Practice", *Byte,* Vol. 16, Jan.. 1991, pp. 267-269.

97. Shifrin, Carole A., "Gate Assignment Expert System Reduces Delays at United's Hubs", *Aviation Week & Space Tech.,* Vol. 128, Jan. 25, 1988, pp. 149-159.

98. Pang, Grantham K.H., "Elevator Scheduling System using Blackboard Architecture", *IEEE Proc.-D,* Vol. 138, No. 4, July 1991, pp. 337-352.

99. Potvin, G. Lapalme, and J. Rousseau, "Integration of AI and OR Techniques for Computer-aided Algorithmic Design in the Vehicle Routing Domain", *J. of Opl. Res. Soc.,* Vol. 41, 1990, pp. 517-522.

100. Michalewicz, Zbigniew. *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1992.

101. Michalewicz, Zbigniew, and C. Jainkow, "GENOCOP: A Genetic Algorithm for Numerical Optimization Problems with Linear Constraints", *Comm. of the ACM*, 1992.

102. Brooke, A., D. Kendrick, and A. Meeraus. *GAMS: A User's Guide*, The Scientific Press, 1988.

103. Michalewicz, Zbigniew, George A. Vignaux, and Matthew Hobbs, "A Nonstandard Genetic Algorithm for the Nonlinear Transportation Problem", *ORSA J. on Computing*, Vol. 3, No. 4, 1991, pp. 307-316.

104. Michalewicz, Zbigniew, "A Genetic Approach for the Linear Transportation Problem", *IEEE Trans. on Sys., Man, Cyber.*, Vol. 21, No. 2, 1991, pp. 445-452.

105. Brown, Donald E. and Christopher Huntley, "Freight Routing and Scheduling at CSX Transportation", *Interfaces* , forthcoming.

106. Wright, M. B., "Applying Stochastic Algorithms to a Locomotive Scheduling Problem", *J. of Opl. Res. Soc.*, Vol. 40, No. 2, 1989, pp. 187-192.

107. Alfa, Attahiru, Sundresh Heragu, and Mingyuan Chen, "A 3-Opt Based Simulated Annealing Algorithm for the Vehicle Routing Problem", *Computers & Ind. Eng.*, Vol. 21, No. 1-4, 1991, pp. 635-639.

108. Potvin, Jean-Yves, Yu Shen, and Jean-Marc Rousseau, "Neural Networks for Automated Vehicle Dispatching", *Computers Ops. Res.*, Vol. 19, No. 3/4, 1992, pp. 267-276.

109. Nygard, K. E., P. Juell, and N. Kadaba, "Neural Networks for Selecting Vehicle Routing Heuristics", *ORSA J. on Computing,* Vol. 2, No. 4 (Fall), 1990, pp. 353-364.

110. Lin, Feng-Tse, Cheng-Yan Kao, and Ching-Chi Hsu, "Applying the Genetic Approach to Simulated Annealing in Solving Some NP-Hard Problems", *IEEE Trans. on Sys., Man, and Cybernetics* Vol. 23, No. 6, 1993, 1752-1766.

111. Asad, A. and E. Wasil, "Project Management using a Microcomputer", *Computers Ops. Res.*, Vol. 13, pp. 231-260, 1986.

112. Elmaghraby, S.E., *Activity Networks: Project Planning and Control by Network Models*, Wiley, 1977.

113. Moder, J.J., C.R. Phillips, and E.W. Davis, *Project Managment with CPM, PERT, and Precedence Diagramming*, 3rd ed., Van Nostrand Reinhold, 1983.

114. Wiest, J.D. and F. K. Levy, *A Management Guide to PERT/CPM with GERT/PDM/DCPM and Other Networks*, 2nd ed., Prentice-Hall, 1977.

115. Gupta, S.K. and L.R.Taube, "A State of the Art Survey of Research on Project Management", in *Project Managment: Methods and Studies*, Elsevier Science Publishers, 1985.

116. Hendrickson, C. and B.N. Janson, "A Common Network Flow Formulation for Several Civil Engineering Problems", *Civil Eng. Sys.* Vol. 1, June 1984, pp. 195-203.

117. Falk, J.E., and J.L. Horowicz, "Critical Path Problems with Concave Cost-Time Curves", *Mgmt. Sci.*, Vol. 19, 1972, pp. 446-455.

118. Patterson, J.H., "A Comparison of Exact Approaches for Solving Multiple Constrained Resource Project Scheduling Problems", *Mgmt. Sci.*, Vol. 30, July 1984, pp. 854-867.

119. Stinson, J.P., E.W. Davis, and M. Khumawala, "Multiple-Resource Constrained-Scheduling using Branch and Bound", *AIIE Trans.*, Vol. 10, 1978, pp. 252-259.

120. Willis, R.J., and N.A. Hastings, "Project Scheduling with Resource Constraints using Branch and Bound", *J. of Opl. Res. Soc.,* Vol. 27, 1976, pp. 341-349.

121. Petrovic, R., "Optimization of Resource Allocation in Project Planning", *Ops. Res.,* Vol. 16, 1968, pp. 559-567.

122. Crowston, W., and G.L. Thompson, "Decision CPM: A Method for Simultaneous Planning, Scheduling, and Control of Projects", *Ops. Res.*, Vol. 15, 1970, pp. 435-452..

123. Hindelang, T.J., and J.F. Muth, "A Dynamic Programming algorithm for Decision CPM Networks", *Ops. Res.*, Vol. 27, 1979, pp. 225-241.

124. Davis, E., and J.H. Patterson, "A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling", *Mgmt. Sci.,* Vol. 21, 1975, pp. 944-955.

125. Kurtulus, I., and Davis, E., "Multi-Project Scheduling: Categorization of Heuristic Rules Performance", *Mgmt. Sci.,* Vol. 28, 1982, pp. 161-172.

126. Zong, Y., T. Yang, and Ignizio, J.P., "An Expert System using an Exchange Heuristic for the Resource Constrained Scheduling Problem", *Expert Sys. with Applications,* Vol. 8, No. 3, Jul.-Sep. 1993, pp. 327-340.

127. Zozaya-Gorostiza, Carlos, Chris Hendrickson, and Daniel R. Rehak, *Knowledge-based Process Planning for Construction and Manufacturing, Academic Press, 1989.*

128. Barber, T.J., and J.T. Boardman, "Knowledge-Based Project Control Employing Heuristic Optimization", *IEE Proc. Part A*, Vol. 135, No. 8, 1988, pp. 529-538.

129. Barber, T.J., J.T. Boardman, and N. Brown, "Practical Evaluation of an Intelligent Knowledge-Based Project Control System", *IEE Proc. Part A*, Vol. 137, No. 1, 1990, pp. 35-51.

130. Lawton, George, "Genetic Algorithms for Schedule Optimization", *AI Expert*, May 1992, pp. 27.

131. Hou, Edwin, S., Nirwan Ansari, and Hong Ren, "A Genetic Algorithm for Multiprocessor Scheduling", *IEEE Trans. on Parallel and Dist. Sys.*, Vol. 5, No. 2, 1994, pp. 113-120.

132. McKim, Robert A., "Neural Network Applications for Project Management: Three Case Studies", *Proj. Mgmt J.*, Vol. 24, No. 4, 1993, pp. 28-33.

133. Adorf, H.M., and M.D. Johnston, "Scheduling with Neural Networks — The Case of the Hubble Telescope", *Computers Ops. Res.*, Vol. 19, No. 3/4, 1992, pp. 209-240.

134. Adorf, H.M., and M.D. Johnston, "A Discrete Stochastic Neural Network Algorithm for Constraint Satisfaction Problems", *Proc. Int. Joint. Conf. Neural Networks Vol. III*, San Diego, CA, 1990, pp. 917-924.

135. Kumar, S., B. Krishna, and P. Satsnagi, "Fuzzy-Systems and Neural Networks in Software Engineering Project-Management", *Applied Intel.*, Vol. 4, No. 1, 1994, pp. 31-52.

136. Simon, Herbert A., and Allen Newell, "Heuristic Problem Solving: The Next Advance in Operations Research," *Ops. Res.*, January-February 1958, pp. 1-10.

137. Kemph, Karl, Bruce Russell, Sanjiv Sidhu and Stu Barrett, "AI-Based Schedulers in Manufacturing Practice: Report of a Panel Discussion," *AI Mag.*, Vol. 11, Jan. 1991, pp. 37-46.

Schedulers & Planners:
What and How Can We Learn From Them

Kenneth N. McKay[1], Frank R. Safayeni[2], and John A. Buzacott[3]

[1]*Faculty of Business Administration, Memorial University of Newfoundland, St. John's, Newfoundland, A1B 3X5, Canada,*
kmckay@kean.ucs.mun.ca

[2]*Dept. of Management Sciences, University of Waterloo*

[3]*Faculty of Administrative Studies, York University*

Abstract

As part of a larger project investigating the nature of planning and scheduling, we conducted an interdisciplinary research agenda focusing on the **scheduler** and the **scheduling task**. Preliminary field research resulted in a paradigm being derived for planning and scheduling in industries that are under a state of development or that have a degree of inherent uncertainty (e.g., electronics, ceramics, biotech, and custom job shops). Methods from the social sciences (e.g., Organizational Behavior and Cognitive Science) were then used to gather and analyse information from the field to validate and support the concepts. This paper describes the approach taken for the field research and discusses the benefits that can be obtained when a social science perspective is taken to studying the management of operations.

Key words: interdisciplinary research, field methodology, production scheduling

1. Introduction

One of the pleasures of conducting field research on planning and scheduling is the discovery of "gems". For example, we have observed several schedulers consciously assigning work to the least flexible resources first which is similar to a heuristic studied by Yao and Pei (1990) but with a different purpose in mind. The schedulers were protecting the most powerful (flexible) resources for future reactive measures while Yao and Pei were using flexibility as a way to generate "quality" schedules. Thus, one could speculate that the schedulers were not necessarily making suboptimal decisions (by accident they might have been doing something as good as shortest-processing-time (SPT)); and one could speculate that the Yao and Pei approach may be more reasonable to consider than SPT in certain real world situations for reasons beyond that of schedule optimality. Although we were not looking for this type of information, we think an interesting insight was possibly obtained.

Is the above story yet another delightful scheduling ancedote without scientific rigor? Yes, it is an interesting story. Unfortunately, this type of story is not a rarity and we suspect it is similar to the majority of other knowledge about manufacturing logistics gained through field research -- interesting stories with no scientific support. As a growing number of researchers are going out into the field to perform applied research, we must be careful about the knowledge we are gathering and when possible elevate the stature of the knowledge beyond that of being a mere story. We believe that researchers attempting to study the *scheduler rather than scheduling* need to be aware of the various social science methods and not rely on ad hoc data gathering and storytelling. This awareness of the social science methods is the focus of this paper.

We observed our flexibility "gem" during one of our earlier field projects that involved a number of case studies (McKay 1987). It was one of many interesting heuristics that were observed and documented as isolated items, each possibly worthy of scientific exploration. In this paper, we will use the phrase "heuristic" to refer to any rule of thumb or approximation technique which is used to make a decision - specifically at the dispatch or work release level. We also noted heuristics about the decision process itself and how the decision process should change based on the perceived state of the factory and work load. In one sense, our initial work explored and documented the various heuristics used

by the human scheduler (e.g., why was a certain heuristic used in a specific situation and not in another).

When the next phase of our scheduling research began, we decided that we would attempt to take our scheduling research beyond the storytelling mode and into the realm of quantitative measurement. We wanted to be able to place individual heuristics in context and to study the semantic composition of the heuristics. We also wanted to investigate some preliminary concepts about scheduler skill and expertise. Following a short overview of our research agenda, we will describe the field methodology, analysis approach, and the types of "supportable" observations we were able to obtain. The paper concludes with a discussion about some of the issues involved with performing research on schedulers and the scheduling task.

2. Background

Conventional research on planning and scheduling has focused on selecting the best schedule from a feasible set using one or more objective functions and a dozen or so important characteristics about the scheduling problem (e.g., processing times, operation steps, travel time, queue sizes, and setups). The feasible set is typically defined using physical properties: material is available, machine is capable, machine is available, and there is a need for the part to be made or processed. It is also possible to include other characteristics such as: operators, tooling, preferences, yields, machine breakdowns, operation sequences, fixtures, and material handling considerations to the feasibility criteria, but the philosophy remains that of physical feasibility. This basic approach makes sense -- it establishes feasibility first, and then optimizes within the bounded region. The approach also makes sense for a number of real world situations where the assumptions of physical feasibility and a few selected manufacturing characteristics are sufficient for determining plans and schedules. For example, in many traditional mass production settings (i.e., process industries, flow shops, repetitive job shops) the "academic" problem definition is valid and many results have been transferred to live settings.

Why hasn't the same phenomenon of technology transfer occurred for other manufacturing situations such as custom job shops with high mix and low volumes, or high technology flow shops? In these various industries which resemble dynamic job shops, there have been few reported research results that have been applicable to

"general" situations or that have been transferred to industry (e.g,. Becker 1985, Buxey 1989, Rodammer and White 1989, Kempf et al 1991). This is at the same time as industry continues to struggle, looking for ways to improve productivity and efficiency (e.g., Dertouzos et al 1989, Klein 1990).

- Is it because planners and schedulers do not appreciate the benefits of our research? -- this assumes that our results are correct and the fault for not using them lies in the hands of the targeted user.

- Is it because the rest of the manufacturing system does not support the scientific approach to factory coordination despite the scheduler's desire and understanding? -- this view assumes that the research is applicable, decision maker is appreciative, but the infrastructure does not exist for its deployment.

- Is it because the academic results are neither feasible nor optimistic in a realistic sense? -- this assumes that significant aspects of the real problem have not been captured and modelled.

There are of course other questions and issues, but where does one start and how does one avoid mere storytelling?

We decided to start at the beginning and by using a hybrid behavioral and cognitive science approach, try to answer the seemingly simple question: *What is scheduling in the real world?* We felt that this social science approach would help us understand the relationship between theory and practice.

The first phase of the research involved a number of short case studies that looked at the scheduling task from an Organizational Behavior perspective. This included the documentation of information flows and analysing what tasks the scheduler performed. The research resulted in: i) a conceptual framework for viewing the scheduler's role, and ii) a number of insights about scheduling practice. Many different types of tasks were documented and numerous scheduling constraints were also identified (McKay 1987).

Three observations made during the first phase inspired the research direction described in this paper:

- Scheduling is a complex task involving many types of decisions and activities, one of which is assigning work to machines using physical feasibility criteria;

- Schedulers keep track of exceptions and changes to their environment -- any change or anticipated change implies potential risk;

- Superficially, the decision process called scheduling seems to be like an amoeba - without a constant structure, shape, or form. The constraints, objectives, and decision rules constantly change and adapt to the changing environment in which the decision process functions.

Based on the first round of empirical research and inspired by the above observations, research began on a dynamic and adaptive paradigm for planning and scheduling when uncertainty is present in many forms. In a spirit similar to Anthony (1965), we view preliminary paradigms as useful for structuring and guiding exploratory research. The paradigm may turn out to be insufficient or invalid, but it can still serve a valuable purpose. Without going into the details, the paradigm that was derived can be considered a variant of the Anthony (1965) concept with accommodation for issues associated with rapidly changing industries. These issues and their affect on the mid-level of decision making were briefly noted by Anthony (1988). We specifically focused on the role of informal information and human judgement at the short and mid term decision levels. This implied the need for an interdisciplinary approach to the research activity -- in the modelling concepts and in field experimentation. The approach taken for field experimentation and what fundamental knowledge was obtained will be discussed in the following sections.

3. Field Methodology

Field experimentation became a major component of the research agenda. This is not to say that field studies were the first or only option considered -- in fact they were the last thing we wanted to do. While we enjoyed visiting factories and talking to schedulers, we knew that the planning and scheduling task was a messy and

complex domain and it would be difficult to obtain quality data which could be used to support any hypothesis or claim. Many discussions took place about the possible design and validity of laboratory experiments, but it was difficult to conceive of a controlled experiment that would be a reasonable study of scheduling skill and knowledge. In the end, it was decided to study the subjects in their natural environments and to use a Cognitive Science perspective for eliciting, transcribing, and analysing the field data. There have been many cognitive skill studies on topics such as chess and solving physics problems (Chase and Simon 1973, Simon and Chase 1973, Larkin et al 1980, Chi et al 1981, Chi et al 1982) and this research base was used to devise an approach for studying schedulers.

When the majority of the decision tasks are performed by an individual, it appears appropriate to view the situation as an integrated process and study the individual using techniques designed to capture thought processes. However, there are many instances where several people share the tasks and it is awkward to study this type of situation using methods designed for probing individual behavior. We devised methods for each of these situations and also devised an approach for transcribing a subset of the gathered data to a common form for the purpose of analysis. The two styles of field study will be described in the following subsections. A complete description of the field instruments and encoding scheme can be found in McKay (1992).

3.1 Integrated Decision Making

One of the field studies had a focused factory concept with a single individual responsible for the majority of short to mid term work assignment decisions. This individual was able to be carefully tracked for reactive, daily, weekly, and monthly decisions. In this case, it was possible to study individual decisions, predictions, results of actions, heuristics used, types of information used, accuracy of predictions, and so forth.

One researcher performed the field study and worked closely with the scheduler. The researcher had an extensive background in electronics manufacturing and understood the underlying technology, manufacturing process, terminology, and procedures used by the company. The scheduler noted that he had time to explain the scheduling, but not to train someone about electronics manufacturing or corporate life. A one-to-one approach was

specifically used instead of a team - we had to establish a bond with the scheduler since he had to share his secrets and trust us. This was crucial since some of the information from the scheduler did not come forth until after the researcher's integrity had been "tested" and the scheduler believed that the researcher was actually trying to improve the situation. Some of the information provided by the scheduler was provided in confidence and was not disclosed to his management, co-workers, or in the research reports. Although it was not done in this study, another researcher at the site would have added robustness and rigour to the study. The second observer would not have been used for interacting with the scheduler (this would have created unwanted dynamics), but would have been used for obtaining additional background data, checking facts, validating claims, and so forth.

A number of techniques were used by the researcher to establish the necessary rapport and overcome the academic and management stigma. Some of the techniques were as simple as being there before he arrived at seven in the morning and leaving when he did at six or seven at night, or going out for smoke and coffee breaks with him and his friends in the winter cold (although the researcher did not smoke). An effort was also made in the early stages to listen and observe and not offer advice and provide quickly thought up solutions. After a suitable period of time, it was appropriate to start discussing the situations (perhaps using additional research data) and asking for his views and his help sorting out the issue under discussion. The scheduler being studied had no formal training and had only been scheduling for nine months, but he had over fifteen years of experience in the factory and this experience had to be respected.

The field study was performed over a six month period during which there was one intensive field visit (five days of one week) and then weekly visits for nineteen weeks. The longitudinal design was chosen to provide multiple sample points across several manufacturing life cycles: fiscal months, quarters, year end, new products, line changes, machine upgrades, new processes, and altered procedures. As part of our earlier case studies we have observed significant changes in the decision process over time. The constraints, objectives, and scheduling heuristics used by the decision makers would appear to vary from visit to visit and would also vary depending on when the assignment was being contemplated for on the planning horizon. In this study we decided

to capture and study the variability in the decision process explicitly.

A preliminary questionnaire was used to profile the decision maker, direct observations were made during the first week, and then a structured probe was used for each of the weekly visits. The weekly probes took approximately one hour and were conducted during the scheduler's weekly planning activities. An additional two to three hours were spent at the site each week to cross check information. The structured probe was designed to capture the manufacturing aspects the scheduler was concentrating on. For example, the scheduler was probed on material, machine, current products, new products, personnel, etc. to see if there was anything special going on that he had to attend to. Each "report" became an episode for later transcription and analysis (Olson and Biolsi 1991). Over the study period, 241 episodes and 67 predictions were captured via this technique (McKay 1992).

A frequently used technique for performing field studies is the taping of sessions. This technique is invaluable in some situations, but can be a drawback when the subject is suspected of having a "multiple personality" and the researcher does not want to inhibit the subject. We believe that schedulers maintain multiple fascades and have to be dealt with carefully. For example, we have heard many managers describe how the scheduler makes decisions; heard a similar story from the scheduler in the first few meetings; but then have the story **completely** change when the scheduler felt comfortable with us and had taken us into his or her confidence. We believe that if the schedulers were taped, the latter versions would never come out -- schedulers are frequently blamed for everything by everyone and appear to be very suspicious of strangers. It has always taken several visits or several weeks before the decision makers have trusted us with confidential information about their decision making. Thus, we relied on manual note taking, summarizing the data at the end of the day, and then cross checking the data with both the scheduler and others. This can introduce inaccuracies and incompleteness in the data which has to be taken into account during analysis. We segregated predictions and related data into ambiguous and unambiguous categories to clearly identify "strong" evidence. A conservative approach was also used to down rate any observation within a coding scheme -- thus ensuring any reported findings would be conservative and not optimistic. We further used a four point scale on the prediction data

to force polarization and to avoid mid-group bunching; forcing data to be treated even more conservatively.

After the raw data describing the episode was collected, each episode was analysed and encoded using a standardized codex. The encoding approach is fairly generic and consists of reviewing the data for evidence of certain characteristics or data. The encoding scheme used for the episodes was:

Summary Data:

- *Position* - if the decision was at the planning, mid-level, or last minute reactive level
- *Outcome* - if the decision affected the internal decision process, the existing schedule, the physical system, or was merely a status update
- *Focus* - (a refinement of the outcome) - if the decision affected the declarative or procedural information, triggered special analysis, or affected the capacity, process, or procedures
- *Temporal* - when the result of the decision can be seen in the factory - immediately, ongoing, or at some point in the future
- *Key Words* - phrases or terms from a lexicon which could be used to isolate or categorize the episode

Predictions noted in an episode were encoded:

- *Focus* - if the prediction was focusing on capacity, demand, or capacity/demand
- *Accuracy* - if the prediction was considered to be ambiguous, accurate, wrong (and degrees of each)
- *Horizon* - when is the prediction for - immediate, future, or sometime in the future after a certain event happens
- *Trend* - if the prediction appears to use information that is cyclic or based on repeated behavior
- *Computer Encoding* - if the prediction uses information which can be captured and maintained in a computer system for automated use

- *Span* - length of time between the time of prediction and outcome

- *Information Enrichment* - if the prediction uses information not normally found in computerized or formal information systems

- *Self* - if the scheduler made a prediction about his own performance

- *Second-Guess* - if the scheduler was second guessing a superior, engineering expert, etc.

- *Surprises* - if the episode being documented was a surprise to the scheduler

Heuristics observed in an episode were encoded:

- *Frequency* - if the heuristic was expected to be seen daily, weekly, in reactive situations, or in predictive planning

- *Trigger* - what appears to activate the heuristic and makes it applicable - characteristics relating to resource, organization, product, process

- *Focus* - if the heuristic was focusing on risk avoidance, impact minimization, moving other work out of danger's way, creating capacity

- *Constraint* - if the heuristic relaxed constraints

- *Scope* - if the heuristic was generic to many manufacturing situations or strictly to the site being studied

- *Visual* - if the heuristic would have a visual impact on the work sequence or work release - e.g., would the schedule be updated

- *Time* - if the heuristic had some time dependency or sensitivity

- *Decision Process* - if the heuristic affected the decision process - analysis, procedural information, input data, output expectation

- *Systemic Concept* - if the heuristic had some notionof systemic themes

- *Wisdom* - if the heuristic had what would be considered general or specific wisdom

- *Adaptation* - if the heuristic adapted the decision process or external environment around the scheduler

- *Decision Level* - if the heuristic was working at the planning, mid-level, or last minute reactive level

- *Information Enrichment* - if the heuristic used information not normally included in manufacturing information systems

- *Computer Encoding* - if the heuristic used information or logic which could be readily captured or used in a computerized system

The encodings were then used to view the collection of episodes. For example, one of the types of data collected referred to surprises encountered by the scheduler -- things he was not expecting. For example, there were 88 perturbations during the study period that were considered important enough that the production targets were threatened. Of the 88, 21 caught the scheduler completely by surprise; the situation happened and the scheduler was forced to react to the situation after the impact occurred and played its course. This type of systematic encoding allows for the episodes to be analysed by computer programmes - enumerating instances or combinations across the episodes.

It is obvious that this approach concentrated on the individual and draws its strength from studying the single subject intensely. This is also the method's weakness when multiple people are involved constantly in the decision process. The following subsection describes the methodology used to study the group decision process.

3.2 Distributed Decision Making

Situations with distributed decision making, such as organizations with a formal decision hierarchy, are difficult to study with the above method. It is possible to track each of the members of the decision making team from a skill and expertise perspective, but it

is not clear how multiple samples can be integrated. There can be additional problems when group decision making occurs at morning meetings -- hundreds of work orders can be reviewed in a short time interval and time to gather information about each decision is limited. One of the study sites had a distributed decision process and a different approach to gathering decision information was derived.

For this study, we used one researcher/observer who worked with the manager in charge of scheduling and this provided a natural focal point for obtaining status and background information about the decisions. During the morning meeting, the observer would sit in an corner and document the special decisions and note any aspects that needed additional information or substantiation. The rest of each site visit consisted of background research and understanding the decisions which were made at the morning meeting. In our group setting, the decisions were typically discussed via a dialog that went like: there is a certain problem, what was supposed to happen, what are the possible options, and what has to happen to make the desired option *real*. This style of dialog seemed to fit the action-goal-trigger perspective (Von Cranach and Kalbermatten 1982).

This perspective was used to view abnormalities in the decision process. Special decisions were documented as to the action taken; what appeared to trigger or facilitate the decision; and what was the primary goal of the action. For example, moving a compatible part from a test fixture to a potentially late order could be viewed as:

- Action (borrowing parts)
- Goal (meet due date)
- Trigger (similar part found elsewhere in plant)

These triples of Action-Goal-Trigger components form a decision combination that can then be reviewed and encoded in a fashion similar to heuristics observed in episodes. For example, they can be encoded as to the generality of the decision (e.g., specific to the factory and products being made, or applicable to other firms); what type of information was used; if it had a short term effect; and so forth. The specific encoding options were:

- *Adaptation* - was the change (if any) being made to the decision process or factory situation a short-, mid- or long-term change

- *Decision Level* - was the decision at the planning level, mid-level, or of a last minute reactive nature

- *Scope* - was the decision was considered generic to manufacturing in general or was specific to the site

- *Enrichment* - was the decision combination based on information normally found in manufacturing information systems, or was it based on other information

- *Computer Encoding* - was the decision combination based on information that could be captured and used in a computer system (partially or in its entirety)

- *Key Words* - one or more words from a lexicon that would capture the key aspects of the decision combination

We tracked the unique combinations that appeared during the study and did not attempt to track the number of times a combination occurred. Over the five month period, a total of 54 unique non-routine decision combinations (Action-Goal-Trigger) were observed in the group situation. This was based on data from thirteen group meetings which were monitored and documented. Each meeting dealt with a status update on approximately two hundred work orders (about to be released or on the factory floor). The decision combinations reflected the special actions taken to correct or adjust any potentially "late" work. At least ten percent of the items would require some kind of decision/action per meeting and there were times the percentage was quite a bit higher. The 54 combinations reflect the population from which the decision was taken - all of the special decisions observed mapped to one of the combinations.

Although we did not gather data on this aspect, it almost appeared as if the decision makers had a portfolio of possible *action-reaction* decisions which they had used in the past. When a problem was discussed, they seemed to map each day's issues into the subset - the organization's shared memory. This makes some intuitive sense since the amount of effort to arrive at decisions would be lessened and the individuals involved would know the necessary actions with a minimal amount of communication. It would have been

interesting to see how the group learned or accepted new combinations or adapted to new situations. It would have also been interesting to see how often and when a suboptimal solution was taken because it was already known by the group versus the testing of an unproven solution.

We found that this group study approach was not as informative or as rich as the individual episode approach taken for the integrated decision process. Although the same type of analysis was used for both heuristics and decision combinations, the episodic approach yielded far more information. To upgrade the distributed methodology would require many researchers on site (one per decision maker) as well as a better method for dealing with group decision situations. For example, it would be necessary to track the individuals before and after the meetings and map the pre and post activities to the dialog during the meeting. In our group study, there were over eighteen people directly involved at the morning meeting - each having two to three assistants providing status information. It is obvious that this type of decision situation would require an immense investment in resources if it were to be studied via a method based on each individual's decision making.

Despite the inherent shortcomings, the approach taken for studying distributed decision making still allowed us to document when certain types of decisions occurred in the manufacturing process and where flexibility existed in the decision process. For example, the factory we studied with the group method fabricated and assembled end products that were customized or tailored to each customer with varying amounts of manufacturing resources needed to make the product. In this factory, almost all decisions were routine with no sensitivity to mix or dynamics until almost the last minute before the product was shipped. While we do not agree with all of the practices we observe in the real world, we feel that it is still important to be able to capture and represent the phenomenon for the purpose of analysis and discussion.

4. Information Analysis

Depending on the stage of research in which one finds oneself, different forms of analysis are appropriate. In the beginning, it might be desired to show mere existence. Subsequently, one can proceed to population samples and claim representativeness and normality. In general, the stronger the claim, the more data is

required and even analysing a small amount of field data that deals with decision making can be daunting. The techniques of protocol analysis (Ericsson and Simon 1984) and episodic interpretation (Olson and Biolsi 1991) are designed for rigorously dealing with field data and are well documented, but they do require substantial investments of time and energy.

In the early stages of original research, it is important to be able to describe and explain the observed phenomenon prior to making predictions about what can be expected to be found in another "random" sample. We found our field methods suitable for the early stages of scheduling research when coupled with a simple counting and classification approach. Each episode was evaluated for its semantic content, what elements of the manufacturing were represented, what was being emphasized etc. Then, it was simple to enumerate as a frequency and percentage the number of episodes exhibiting the concepts or points that were encoded. For example, counting the number of episodes exhibiting an awareness of politics, or the number of heuristics that explicitly relaxed or violated constraints. As our research was exploratory, caution was used during the analysis to ensure that simple enumeration was used to show trends and existence -- not to prove statistical significance or claim causal relationships. This counting and classifying approach matches several elementary statistical tests and the tests can show the possible existence of bias and the chance of random observation (Nachmias and Nachmias 1987).

Our research used two longitudinal field studies to probe the scheduling phenomena related to risk and uncertainty -- sufficient to show existence, but insufficient for strong claims. The studies would have to be replicated many times and in many factory situations prior to strong claims being validated. Furthermore, there are many decision making topologies - how the organization is orchestrated to make decisions -- and we only probed two forms: integrated decision making in a focused factory situation and a formal hierarchical system. Before we can make general claims, many other forms will have to be studied. These issues are not unique to our specific use of field methods and general discussions of the issues can be found in many sources (e.g., Glazer 1972, Yin 1989, Jorgensen 1989).

5. Research Findings

The previous sections focused on the methodology - what, how, and why. This section discusses types of research results we have obtained.

As noted in the background section, the first round of field work used the case study approach and yielded: i) a framework for viewing (describing and studying) the scheduling task, ii) a collection of objectives, constraints, and scheduling heuristics, and iii) an initial description of how the scheduling task changes and adapts to perceived risks. The case study approach is relatively informal and is not structured for information analysis. However, it can provide many insights into the situation being studied. The first studies provided the material for preparing a decision paradigm for rapidly changing situations.

The second round of field work used methods designed for longitudinal field studies focusing on a specific aspect - methods which are significantly different from the case study approach. We used the field studies to explore and test a number of hypotheses about the derived decision paradigm.

The longitudinal field study used a Cognitive Science perspective and showed that it is possible to conduct a quantitative analysis of the scheduling process (e.g., classifying, counting) by carefully enumerating many qualitative aspects of the scheduling process (e.g., informal information, political influence).

In addition to being able to "test" various parts of the decision model without relying on simple stories or examples, the field study approach illustrated the types of knowledge a researcher can learn from a human scheduler: knowledge about what the scheduling problem is, knowledge about how schedules are generated, and knowledge about what constitutes a feasible and acceptable schedule in a real setting. We were also able to extract specific items of knowledge about the scheduling situations being studied which in turn provided more general insights:

- We were able to document where the information came from, how it was obtained, under what condition the information was used, and how often it was important enough to be considered. This type of information is important if one is to design expert systems (the types of rules to include),

information systems (what should be gathered), other computer support systems, or if one is making recommendations for changing the scheduling practice in the factory. For example, the one scheduler considered any change made to the manufacturing system (materials, process, procedures, operator training, etc.) since the last time a product was built. This information was used to determine batch splitting, test runs, degree of confidence in engineering estimates, and so forth.

- We were able to isolate decisions about the decision process versus the resulting schedule. This provided insights into adaptive scheduling systems and how the decision logic itself must sense certain triggers and adjust. For example, during certain periods of the month, there would be changes in the options allowed the scheduler - some political, some determined by vendors, some by the state of the product's maturity, etc. These types of options would change daily or weekly and affect the decision process. The scheduler also adjusted the schedule (work assignments) based on information such as the operators' attitude during training.

- We isolated a number of heuristics for sensing points in the future where problems could be expected and a corresponding set of heuristics for adjusting the work assignment in anticipation of the problem. These specific heuristics try to avoid dumb decisions and have suggested a class of general heuristics for dynamically adjusting the priority of work based on anticipated problem points and moving certain work away from the aversion point or moving other work toward the point (to absorb or test the situation). Work has now started on formal modelling of the heuristics.

- We established a possible technique and measurements for determining scheduler quality and performance. For example, how many problems are anticipated, how successful are the special decisions, and how far into the future is the scheduler accurate. Repeated studies can possibly indicate learning and skill acquisition on the part of the scheduler and sufficient studies could yield information about scheduler performance in general and provide a benchmark for scheduling systems or schedulers.

A formal field study approach also provides the data for in-depth analysis of specific aspects of the scheduling process. Instead of simply taking a *rule of thumb* as given and blindly putting it into a system or algorithm, we can start to understand why a certain decision was made and what lay beneath the surface. For example, consider just one aspect:

- During the one integrated study, we isolated and analyzed 128 heuristics -- capturing the context in which they were used and mapping the heuristics against the decision paradigm. Without a framework and consistent methodology, capturing and analyzing this number of heuristics would have been difficult.

The above concepts and findings were investigated with a fair degree of risk. One of the most important risks of this type of research is the type of knowledge needed by the researcher and how this can impact the study. The knowledge can bias observations, transcription, and interpretation. However, the knowledge is needed to understand what is being observed and what is going on around the single incident being observed. The type of knowledge we are concerned about includes: real world scheduling practices; product and process specifics for the industry being studied; characteristics of the manufacturing system at the site to be studied; and the corporate culture at the site. It would be very difficult, if not impossible, to take a researcher without this knowledge and obtain insightful information.

6. Conclusion

The preliminary investigation provided sufficient empirical support for further development of our decision paradigm and detailed framework. As this was our first study of schedulers using this approach, it was not surprising to discover weaknesses and insights about the method itself. We have documented a number of recommendations (McKay 1992) about the field method and analysis techniques that would improve the approach. In its current state, the approach is applicable for exploratory and preliminary studies at the descriptive level. However, if normative results are sought, a number of changes are necessary to provide for adequate intra and inter sample analysis. We hope to continue the development of this field methodology as our research models evolve into the normative form.

As stated in the introduction, we believe that researchers attempting to study the *scheduler rather than scheduling* need to be aware of the various social science methods and not rely on ad hoc data gathering and storytelling. For example, research ethics, questionnaire design, interview techniques, bias control, data transcription methods, and encoder reliability are all topics that have been studied, documented, and discussed (e.g., Lewin 1951, Glazer 1972, Von Cranach and Kalbermatten 1982, Tversky and Kahneman 1982, Ericsson and Simon 1984, Synodinos 1986, Nachmias and Nachmias 1987, Hoch and Loewenstein 1988, Yin 1989, Jorgensen 1989, Olson and Biolsi 1991). As we discovered, more than one method may be needed and the researcher must be aware of the strengths and benefits of each method -- knowing when to use a method and knowing when not to.

We have found this interdisciplinary approach based on skill and expertise an excellent tool to discover and disambiguate scheduling practice:

- Under what conditions certain decisions are made.
- What are the primary or key aspects of the decisions being made.

Not all of the decision makers we have studied would be considered good or excellent schedulers. Indeed, some clearly exhibited a lack of skill and ability. However, we have always learned something from each of them and it is perhaps best to keep an open mind and consider one of the early descriptions of planning:

> *"Did you ever see a bird, in its search for twigs, straw and the like? Hunting these things for fun? Hardly. It is simply planning ahead against the time when a warm comfortable nest will be wanted for the little ones to come. It does not wait until they have arrived - the bird sees to it that the nest is ready before it will be needed, and, as a result, we call it a wise bird." (Knoeppel 1911, p. 220)*

7. Acknowledgment

The researchers would like to express their gratitude to the various schedulers and planners who have suffered us and contributed to our understanding of the scheduling task. This work was partially

supported by NSERC Grant STR0032731 on Robust Planning and Scheduling. We would also like to thank the referees for thoughtful and useful suggestions.

Bibliography

Anthony, R.N., *Planning and Control Systems: A Framework For Analysis*, Harvard Business School Press, Boston (1965).

Anthony, R.N., *The Management Control Function*, Harvard Business School Press, Boston (1988).

Becker, R.A., "All Factories Are Not the Same," *Interfaces*, 15(3), 85-93 (1985).

Buxey, G., "Production Scheduling: Practice and theory," *European Journal of Operational Research*, 39, 17-31 (1989).

Chase, W.G., and Simon, H.A., "The mind's eye in chess," In *Visual information processing* (W.G. Chase, Ed.), Academic Press, New York, 215-281 (1973).

Chi, M.T.H., Feltovich, P.J., and Glaser, R., "Categorization and Representation of Physics Problems by Experts and Novices," *Cognitive Science*, 5, 121-152 (1981).

Chi, M.T.H., Glaser, R., and Rees, E., "Expertise in Problem Solving," In *Advances in the psychology of human intelligence* (Sternberg, Ed.), 7-75 (1982).

Dertouzos, M.L., Lester, R.K., and Solow, R.M., *Made In America*, Harper Perennial, New York (1989).

Ericsson, K.A., and Simon, H.A., *Protocol Analysis: Verbal Reports as Data*, The MIT Press, Cambridge (1984).

Glazer, M., *The Research Adventure: Promise and Problems of Field Work*, Random House, New York (1972).

Hoch, S.J., and Loewenstein, G.F., "Outcome Feedback: Hindsight and Information," Working paper: University of Chicago, Graduate School of Business Center for Decision Research (1988).

Jorgensen, D.L., *Participant Observation: A Methodology for Human Studies*, Sage Publications, London (1989).

Kempf, K.G., Russell, B., Sidhu, S., and Barrett, S., "AI-Based Schedulers in Manufacturing Practice," *AI Magazine*, 11(5), 46-55 (1991).

Klein, J.A., *Revitalizing Manufacturing*, Irwin, Homewood (1990).

Knoeppel, C.E., *Maximum Production In Machine-Shop and Foundry*, The Engineering Magazine, New York (1911).

Larkin, J., McDermott, J., Simon, D.P., and Simon, H.A., "Expert and Novice Performance in Solving Physics Problems," *Science*, 208, 1335-1342 (1980).

Lewin, K., *Field Theory in Social Science - Selected Theoretical Papers* (D. Cartwright, Ed.), Harper & Row Publishers, New York, (1951).

McKay, K.N., *Conceptual Framework For Job Shop Scheduling*, Unpublished MASc Dissertation, University of Waterloo, Department of Management Sciences (1987).

McKay, K.N., *Production Planning and Scheduling: A Model For Manufacturing Decisions Requiring Judgement*, Unpublished PhD Dissertation, University of Waterloo, Department of Management Sciences (1992).

Nachmias, D. and Nachmias, C., *Research Methods In The Social Sciences - 3rd Edition*, St. Martin's Press, New York (1987).

Olson, J.R., and Biolsi, K.J., "Techniques for representing expert knowledge," In *Toward a general theory of expertise - Prospects and limits* (K.A. Ericsson and J. Smith, Eds.), Cambridge University Press, Cambridge, 240-285 (1991).

Rodammer, F.A., and White, K.P., "A Recent Survey of Production Scheduling," *IEEE Transactions on Systems, Man, and Cybernetics*, 18(6), 841-851 (1989).

Simon, H.A., and Chase, W.G., "Skill in Chess," *American Scientist*, 61(4), 394-403 (1973).

Synodinos, N.E., "Hindsight Distortion: 'I knew it all along and I was sure about it'," *Journal Applied Social Psychology*, 16(2), 107-117 (1986).

Tversky, A., and Kahneman, D., "Evidential Impact of Base Rates," In *Judgement under uncertainty: Heuristics and biases* (D. Kahneman, P. Slovic and A. Tversky, Eds.), Cambridge University Press, Cambridge, 153-160 (1982).

Von Cranach, M., and Kalbermatten, U., "Ordinary Interactive Action: theory, methods, and some empirical findings," In *The Analysis of Action* (M. von Cranach and R. Harre, Eds.), Cambridge University Press, London, 115-160 (1982).

Yao, D., and Pei, F.F., "Flexible Parts Routing in Manufacturing Systems," *IIE Transactions*, 22(1), 48-55 (1990).

Yin, R.K., *Case Study Research: Design and Methods*, Sage Publications, London (1989).

Decision-Theoretic Control of Constraint-Satisfaction and Scheduling

OTHAR HANSSON[1,2] and ANDREW MAYER[1,2]

[1] *Heuristicrats Research, Inc., 1678 Shattuck Avenue, Suite 310, Berkeley, CA 94709.* {othar, mayer}@heuristicrat.com

[2] *Computer Science Division, 571 Evans Hall, University of California, Berkeley, CA 94720*

Abstract

This paper describes DTS, a *decision-theoretic scheduler* designed to employ state-of-the-art probabilistic inference technology to speed the search for efficient solutions to constraint-satisfaction problems. Our approach involves assessing the performance of heuristic control strategies that are normally hard-coded into scheduling systems, and using probabilistic inference to aggregate this information in light of features of a given problem.

BPS, the Bayesian Problem-Solver (Hansson & Mayer 1989, Mayer 1994), introduced a similar approach to solving single-agent and adversarial graph search problems, yielding orders-of-magnitude improvement over traditional techniques. Initial efforts suggest that similar improvements will be realizable when applied to typical constraint-satisfaction scheduling problems. This paper provides an overview of the project, and contrasts it with other operations research and artificial intelligence approaches to search and optimization.

key words: scheduling, decision theory, heuristic search.

This research was supported by the National Aeronautics and Space Administration under contract NAS2-13656.

1. DTS Problem Domain

The Decision-Theoretic Scheduler, DTS, is designed for over-subscribed project scheduling problems. Traditional operations research work in scheduling has produced highly optimized algorithms for restricted problem representations and restricted forms of the objective function. The goal of this work is to attempt the same caliber of optimization on the less restrictive constraint-satisfaction problem representation and for user-specified objective functions.

Primarily, work on DTS has focussed on search control, particularly through the combination of heuristic evaluation functions. As discussed below, this makes DTS a promising approach for new domains in which sophisticated domain-specific heuristic functions have not been developed. Finally, the utility-theoretic basis of DTS' optimization criteria makes it an attractive approach for problems in which complex tradeoffs must be made among competing tasks and expensive real-world and computational resources.

The current DTS effort is specifically targetted toward experiment scheduling on orbiting telescopes. The initial application domain is the Extreme Ultraviolet Explorer (EUVE), operated by the NASA Goddard Space Flight Center and the Center for EUV Astrophysics (CEA) at the University of California, Berkeley (EUVE 1992). The remainder of the paper focuses on simpler testbed problems such as Eight Queens and a textbook Construction Scheduling example.

1.1. CSP Problem Representation

We phrase these scheduling problems in the language of constraint-satisfaction. Formally, a constraint-satisfaction processing (CSP) problem consists of a set of variables together with a set of constraints on the legal values of those variables. The CSP problem is solved when the variables have been instantiated to a set of values that violate none of the constraints. A wide variety of problems can be phrased as CSP problems, including scheduling, graph-coloring, interpretation of visual scenes, etc. (van Hentenryck (1989) provides a survey).

A large class of scheduling problems can be represented as constraint-satisfaction problems, by representing attributes of tasks and resources as variables. Task attributes include the scheduled time for the task (start and end time) and its resource requirements. The primary attribute of resources is availability or accessibility. A schedule is con-

structed by assigning times and resources to tasks, while obeying the constraints of the problem.

Constraints capture logical requirements: a typical resource can be used by only one task at a time. Constraints also express problem requirements: task T_x requires N units of time, must be started within M days of the start of task T_y, and must be completed before a specified date. Both van Hentenryck (1989) and Zweben et al. (1990) provide concise illustrative examples of scheduling problems represented as CSP problems.

We have also used the Eight Queen problem in experiments and demonstrations, as its simplicity helps to highlight the DTS search mechanism. The problem consists of assigning 8 queens to a chess board such that no two queens lie on the same row, column or diagonal. By associating each queen with a row *a priori*, the problem can be simply modeled as assigning each queen (variable) to a unique column (value), under the constraint that no two queens occupy the same diagonal.

2. DTS Overview

There are numerous inadequacies with existing CSP technologies. Existing CSP heuristics, like all heuristic evaluation functions, are imperfect, and exhibit highly domain-specific performance. Although they often provide useful search control advice, they introduce uncertainty into the search algorithms which rely on them. However, CSP problem-solving paradigms do not address this issue because they fail to provide uncertainty models for heuristic information. Consequently, current techniques are forced to pay a large and unnecessary computational price in cases where the heuristic function makes incorrect classifications. Furthermore, the algorithms are doomed to repeat these costly mistakes forever, as there is no learning mechanism designed to improve a CSP heuristic's performance over time.

Existing heuristic functions confuse many different kinds of information. Some heuristic functions estimate the quality of the completion of a partial schedule. Others estimate the difficulty of finding a feasible solution. This confusion results in inadequate guidance for human experts who are charged with developing good heuristic functions. The development of a good heuristic often amounts to little more than parameter-adjustment to improve performance.

The problem of developing heuristics is compounded by the lack of technological developments which allow evidence from multiple heuristic functions to be combined. For this reason, the selection of appropriate

heuristics and problem-solving techniques for any given CSP domain remains a craft despite years of comparative~study.

DTS, which is derived from previous work on BPS, the Bayesian Problem-Solver (Hansson & Mayer 1989), is designed to address these problems. One area of innovation is the *heuristic error model*: a probabilistic semantics for heuristic information, based on the concept of conditional probability in statistical decision-theory (Hansson & Mayer 1990b). Heuristics are interpreted by correlating their estimates with the actual payoffs of problem-solving instances. When a problem is solved, the heuristic error model is updated, adapting it to the problem's specific characteristics. Multiple heuristics are combined by correlating payoffs with a *set* of heuristic estimates (specifically, by computing probability of payoff conditioned on the multiple heuristic estimates). This provides a sound method for combining multiple~heuristics.

This section describes the methodology which forms the core of DTS. In addition to the traditional tools developed for scheduling systems, the DTS approach relies heavily on technologies such as multiattribute utility theory (Keeney & Raiffa 1976, von Winterfeldt & Edwards 1986), Bayesian probabilistic inference (Cox 1946, de Finetti 1974, Savage 1972), information-value theory (Howard 1965, Raiffa & Schlaifer 1961) and Bayesian learning (Duda & Hart 1973, Lee & Mahajan 1988).

2.1. Decisions in Scheduling

DTS employs decision-theoretic techniques to guide the search for feasible and efficient schedules. Decision theory and its central maximum expected utility principle describe methods for making decisions when the outcomes of those decisions are uncertain. A scheduling system is just such a decision-maker. The decisions to be made by a scheduling system include:

- Which portion of the search tree should be explored next?
- Should search continue, or should the current best solution be output to the user?
- If an infeasible schedule must be repaired, which repairs are best?

When considered in isolation, these decisions seem very difficult. In fact, they are difficult to formulate and solve in a sound and efficient manner. Existing search algorithms make these decisions in an *ad hoc* manner. Our approach is to apply the standard principles of rational decision-making (theory of expected utility (von Neumann & Morgenstern 1944)) to

these decisions. Many artificial intelligence researchers have recently turned to decision-theoretic principles in attempting to engineer sound but resource-conscious systems.

2.2. Heuristic Error Models

The fundamental problem with prior work in scheduling is that the semantics of heuristic functions are defined only in terms of performance: heuristics are "magic" parameters that determine the speed of search. Not surprisingly, an expert's effort in development of scheduling systems is often dominated by time spent handcrafting a high-performance heuristic through parameter adjustment. Because the semantics of heuristics are unclear, even the most sophisticated combination and learning mechanisms are limited in their effectiveness.

DTS takes the approach that there are crucial quantities relating to a state in a search tree, i.e., the attributes of the utility function, including the cost of the search performed, whether a solution was found and the attributes which describe the solution quality. If those attributes were known, decision-making would be trivial. In DTS, heuristic evaluation functions are treated as evidence relating to the value of one or more of the utility attributes. We refer to the set of attributes as the *outcome* of that search tree node.

It is apparent that different heuristics serve to measure different attributes of utility (search cost, solution quality, solution probability). For example, a CSP heuristic such as "Most Constraining Variable" is implicitly encoding information about ease of search: a variable which heavily constrains unassigned variables will produce a smaller search tree. This intuition about the heuristic is borne out empirically.

This association between raw heuristic values and utility attributes is referred to as a *heuristic error model*. Briefly, the heuristic error model provides a simple means of infusing domain specific information into the problem-solving process by associating immediately visible features of a state with a belief about the outcome of that state. "Features" of the state S_i are indicated by a heuristic function, $h(S_i)$, and the association with outcome attributes A_i is provided by the heuristic estimate $P(h(S_i)|A_i)$.

Learning Heuristic Error Models

Historically, nearly all heuristic search algorithms have used the face-value principle of heuristic interpretation, i.e., behaving as if these estimates were perfect. As a result, most existing heuristic search algorithms

violate the basic axioms of consistency and rationality in decision-making.

In contrast, DTS gathers statistics to calibrate the heuristic error model over time, as problems are solved. When introducing the system in a new domain, a *prior* probability distribution is fine-tuned based on "training exercises" with representative problems. This calibration process improves DTS performance, tailoring it to the characteristics of real-world problems as they are encountered. When the heuristic function is imperfect, DTS learns a mapping which "corrects" the heuristic to as great a degree as possible. Finally, the DTS learning capability reduces the burden on human experts to produce highly complex heuristics. Their experience can be encoded as a default initial belief, or *prior probability.*

Combining Heuristics

In our initial experiments in scheduling, a primary advantage of the heuristic error model has been the ability to combine multiple heuristics. Search techniques have never offered powerful methods for combining heuristics. A popular approach is to handcraft a composite heuristic which is a linear combination of individual features.

By combining multiple heuristics, DTS isolates measurements of the difficulty and promise of completing potential assignments. Hence, DTS can make use of heuristics which previously have led to inconsistent performance: if there are any easily characterized contexts (in terms of other features) in which the heuristic performs well, DTS will recognize that fact. This context-dependency of heuristic functions has long been recognized in other search applications such as game-playing.

2.3. Use of Heuristic Error Models

The DTS architecture relies on the Bayesian network data structure (Pearl 1988), a recently developed tool for representing and reasoning with probabilistic information. The Bayesian network is used to provide information for decisions such as the most promising region of the search tree to expand next, and the most promising schedule extension or modification to choose next. As described below, Bayesian networks can integrate a variety of information in the service of such decisions, including multiple heuristic evaluation functions and the search tree's topology.

The nodes of a Bayesian network are variables which represent the attributes of the domain. The arcs of the network connect dependent variables, representing relationships among domain attributes. Dependencies

can be due to functional, causal or correlative relationships among variables.

The variables in the DTS Bayesian network are of two types. One type are variables which represent the multiattribute outcomes (e.g., schedule cost, search cost) of legal partial assignments in the CSP problem's state-space. The other type of variables represent the values of heuristic evaluation functions, the primitive feature recognizers of the domain (e.g., the number of remaining values, the degree of the constraint graph). The structure and semantics of the Bayesian network are described in detail by Hansson & Mayer (1991) and Hansson (1994).

The Bayesian network serves as a database for the probabilistic information acquired during search. Common "queries" on this database are of the following form:

- The "next-best-test," or most crucial piece of evidence to gather. In search, this corresponds to the area of the search tree which is most crucial to explore next.
- The conditional probability of a variable instantiation, given the available evidence. In search, such probabilities can be used together with a utility function for maximum-expected-utility decision making, including the choice of task assignments, schedule modifications, etc.
- The most likely instantiation of all variables, given the available evidence. In search, this can be used as a simplified "situation assessment" of the state of the search.

2.4. Utility-Directed Selective Search

DTS employs advanced decision theory techniques to direct its search process. Decision theory, together with the probabilistic inference machinery described above, enables DTS to determine the best portion of the search tree to explore next. In addition to the obvious improvements in search efficiency, this facility improves the flexibility of DTS. By altering the utility function provided as input to the system, DTS may be tailored to trade off increased search time for increases in schedule quality, or to produce schedules with different desirable attributes. For reactive scheduling applications, alterations to the existing schedule can be given negative utility, in which case DTS will avoid them where possible.

The essence of decision-theoretic search control is the realization that there is quantifiable value in the acquisition of information. It should be clear that some pieces of information are more valuable than others. In

addition, the acquisition of information has costs—in scheduling search, this cost is increased computation time. If these computations squander time and other resources, the solution may be found too late to be of use. If these computations are neglected, a poor solution may be found. However, if these computations are chosen wisely, the system will provide high quality solutions despite limited computational resources. Decision theory has spawned a subfield known as *information value theory* which deals with the issue of *deciding*, at a metalevel, what information to acquire in order to make better decisions at the base level.

Decision-theoretic search control thus involves the isolation of decisions that are made in the course of search, and applying the techniques of decision theory to make *rational* decisions at these choice points. The decisions made in heuristic search include choices among possible search tree expansions and possible heuristic evaluations. DTS applies information value theory by using the maximum expected utility criterion to control its information-gathering search.

Such decisions form the basis of a selective search algorithm that explores the search tree in a nonuniform manner so as to find a high quality solution in as little time as possible. In simple terms, rather than being a "depth-first" or "breadth-first" search, DTS exhibits a "highest-utility-first" search behavior, gathering information most relevant to the decisions that must be made.

In addition, because the selective search decisions are based on the global utility function, the system should exhibit "real-time" characteristics (depth-first hill-climbing) and "optimizing" (exhaustive, branch-and-bound search) characteristics depending on the preferences expressed in the utility function. For example, optimizing behavior occurs when search time is free. When more realistic assumptions about search time are incorporated into selective search, a flexible satisficing search arises. Similarly, memory bounds can be handled by selectively discarding low-utility portions of the search tree.

An additional benefit of the decision-theoretic search control is that search control and heuristic information are represented in a declarative manner. As different search spaces (e.g., partial schedules, complete but infeasible schedules, etc.) correspond to different decision problems, DTS can be applied to any search space. In the prototype system, the techniques have been applied to search through the space of valid partial schedules.

Objective Functions

DTS uses multiattribute utility functions to represent user preferences for both solution quality and computational costs. Multiattribute utility theory (MAUT) is a formalized method for quantifying preference relationships among a set of uncertain outcomes. Most other scheduling systems have a single vehicle—the heuristic evaluation function—for representing both search control knowledge and user preferences. This overlap of search control and schedule evaluation makes the task of constructing good heuristic functions more difficult than it needs to be.

For example, the ISIS (Fox 1987) and SPIKE systems (Johnston 1989) employ suitability functions which state the "preferences" of the user as a function of a task and a time assignment. Consider the optimal schedule for a set of tasks. For the SPIKE and ISIS systems, the suitabilty function which best satisfies the user's preferences is that which "encodes" this optimal schedule by peaked 0/1 values. Such a suitability function essentially encodes search control information. Although this is an extreme case, suitability functions and other heuristic evaluation functions must encode both search control and schedule evaluation information. DTS separates heuristic functions and schedule evaluation, yielding a mathematically principled search algorithm, while at the same time simplifying the knowledge-engineering task for the designer of a new scheduling system.

2.5. DTS Version 1.0

The DTS prototype employed a simplified decision-theoretic control mechanism which was adapted to a conventional backtracking search algorithm: this allowed for controlled experiments on DTS *vs.* traditional algorithms.

The only search control decisions made in traditional backtracking systems are the selections of which subtrees of the search graph to explore next. Once a subtree is selected (by selecting the next variable or value), it is explored exhaustively unless a solution is found. Such an ordering problem can be viewed as a decision-tree. Figure 1 depicts the choice of ordering two subtrees A and B. A simple theorem (Hansson & Mayer 1991) shows that the system's expected utility (search time to first solution) is maximized if variables (or values) are ordered by the quantity $P(v)/C(v)$, where $P(v)$ indicates probability of finding a solution in the subtree, and $C(v)$ indicates the cost of searching the subtree (whether or not a solution is found). $P(v)$ and $C(v)$ are attributes of the payoff mentioned above. The experiments described in the next section confirm that

once P(v) and C(v) are learned, this rule outperforms traditional backtracking search algorithms which interpret heuristic estimates at face value. This result indicates that decision-theoretic search-control improves overall system performance. A similar analysis can also be performed for iterative improvement (Hansson & Mayer 1991).

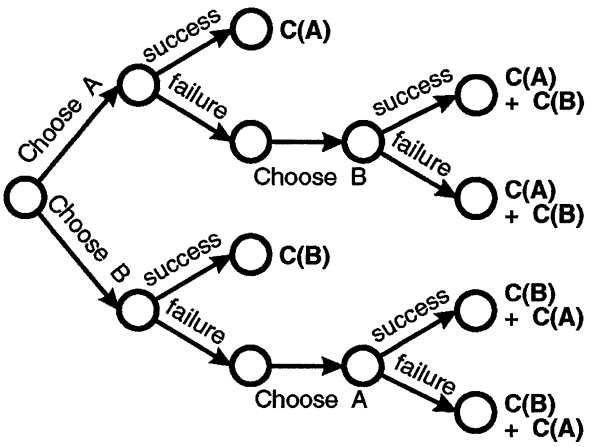


Figure 1. Decision Tree for Value-Ordering Problem (Values A and B)

We note here that while heuristics are usually very good at rank-ordering nodes based on either P(v) or C(v) individually, the rank-ordering for the combination is typically incorrect. DTS' heuristic error model corrects for this.

For clarity, we summarize the prototype implementation here. The prototype performs a backtracking search, using the standard optimizations of forward-checking and dynamic search rearrangement. The search is ordered by the expected utility selection criteria (P(v)/C(v)) discussed above. The estimates of P(v) and C(v) are derived from the heuristic error model, using traditional CSP heuristics. The heuristic error model is learned—i.e., updated during and between trials—using a bucketed histogram together with Laplacian estimation (a standard Bayesian statistical estimator that uses a uniform prior). The buckets in the histogram ensure smooth interpolation, but our results seem to be independent of the bucket size or shape.

2.6. Performance

Space limits us to a discussion of only two aspects of the DTS prototype's performance characteristics: combination of heuristics and learning heuristic error models.

Combining Heuristics

The primary strength of the DTS prototype is the method for combining information from separate heuristic evaluation functions to improve constraint-satisfaction search control. Experiments with the prototype on the Eight Queens and Bridge-Construction Scheduling (van Hentenryck 1989) problems confirm that the combination of heuristic functions provides more information than any of the heuristics taken individually. This translates into significant reductions in overall search time..

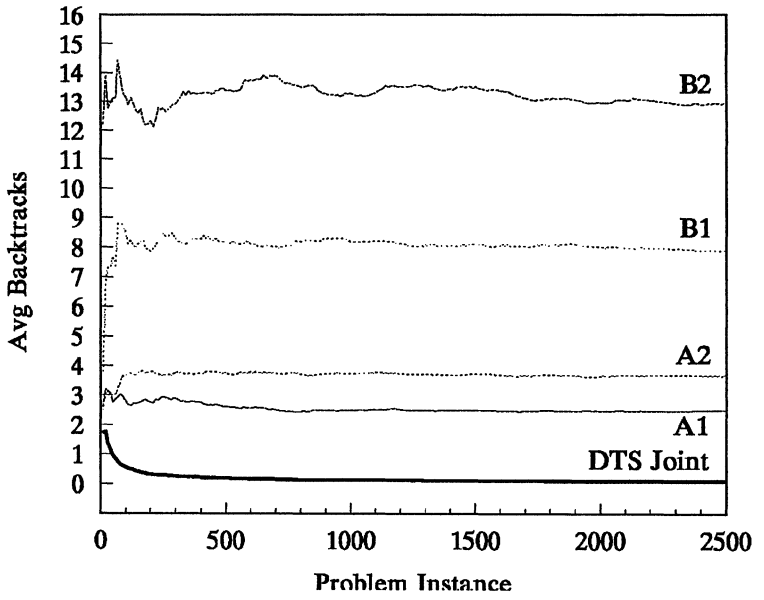


Figure 2. Eight Queens: Combining Heuristics vs. Heuristics in Isolation.

Traditionally, CSP algorithms make use of a variable ordering heuristic and a value ordering heuristic. Figure 2 shows the performance of a standard CSP algorithm using all possible pairings (A1, A2, B1, B2) of two well-known variable ordering heuristics (Most Constraining Variable

(A), Minimum Domain Variable (B)) and two well-known value ordering heuristics (Least Constraining Value (1), Dechter's Value Heuristic (2) (Dechter & Pearl 1988). Also shown is the DTS prototype (DTS-Joint), which dominated the competition by using all four heuristics in combination. The horizontal axis plots the number of problem instances solved and the vertical axis plots the running average of search time over the entire experiment. The plot, but not the average, begins with the tenth problem instance: this is because a running average over small samples has high enough variance to make the graph unreadable (the range of the vertical axis would have to be very large).

Figure 3 shows a corresponding graph for the Bridge-Construction Scheduling problem. The variable ordering heuristic used was Minimum Domain Variable and the value ordering heuristics were Least Constraining Value (curve A1) and ASAP, "as soon as possible" (curve A2). Also shown are the corresponding individual DTS performance curves (DTS A1, DTS A2) as well as the combined heuristic performance curve (DTS-Joint).

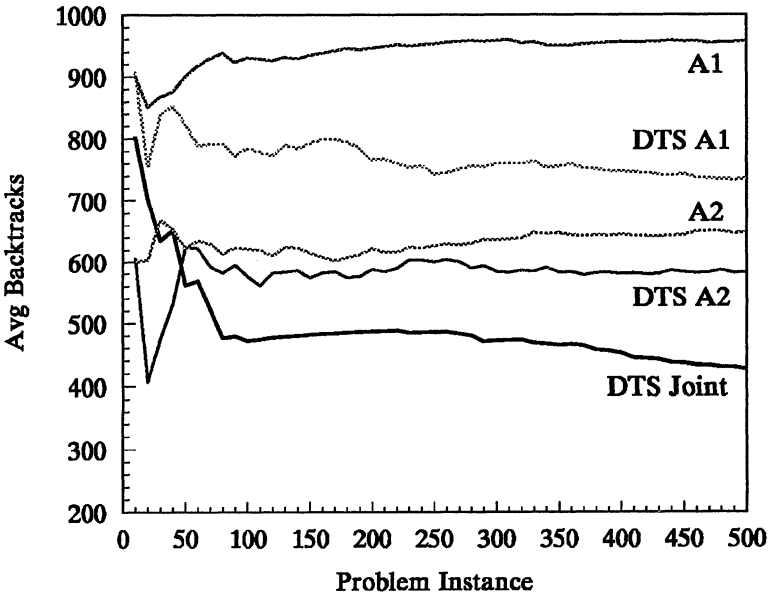


Figure 3. Bridge-Construction Scheduling: Combining Heuristics.

To summarize both graphs, we compare them to the average performance of A1, A2, B1, and B2. The improvement over this average is seen to be nearly 50% for Bridge Construction Scheduling, and over 95% for the Eight-Queens problem. Note that the sharp downward slope of the DTS-Joint *running average* in Figure 3 demonstrates the performance improvement accrued by learning, unattainable using traditional techniques.

Learning Heuristic Error Models

Figure 4 displays an example heuristic error model learned over the course of 2500 Eight-Queens problem instances (for the Minimum Domain heuristic). The horizontal axis plots the heuristic function estimate and the vertical axis plots the preference for that estimate (specifically, the vertical axis is $P(v)/C(v)$). In DTS, preference is based upon the expected utility associated with a heuristic estimate (dashed line). In traditional algorithms, the heuristic is assumed to rank-order alternatives perfectly, and therefore, preference is a monotonic function of the heuristic estimate.

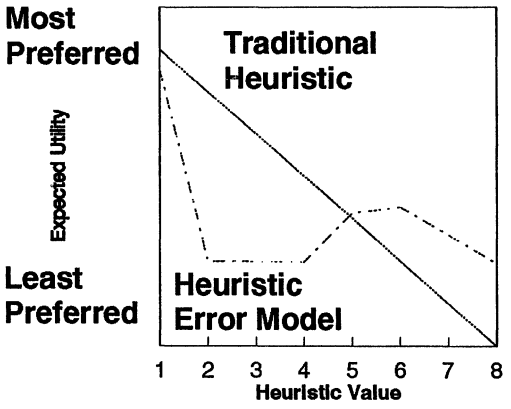


Figure 4. Sample Heuristic Error Model for "Minimum Domain" Heuristic.

The discrepancy between the heuristic estimates and the actual utilities explains the poor performance of traditional approaches, which assume perfect heuristic estimates. Further, it explains why DTS outperforms these techniques, as it does not make this assumption, and instead learns to correct for the discrepancy.

3. Related Work

The DTS system is based on the authors' previous work on the Bayesian Problem-Solver (BPS) system. BPS has been applied to classic AI problem-solving (Hansson & Mayer 1989), game-playing (Hansson & Mayer 1990a) and planning (Hansson et al., 1990) domains. Similar decision-theoretic approaches are being considered for application to other complex multiattribute optimization problems.

This work is most closely related, in assumptions and techniques, to the recent work in applying decision theory to problems such as medical diagnosis (Heckerman 1991) and image interpretation (Ettinger & Levitt 1989). Others have applied decision theory to heuristic search applications. These researchers have typically limited themselves to grafting decision-theoretic principles onto existing algorithms (Lee & Mahajan 1988, Russell & Wefald 1989). Like the decision-theoretic backtracking discussed above, this makes for an interesting starting point, and there are many more interesting possibilities to explore in the future.

Other Optimization Approaches

The explicit use of utility resembles approaches which use objective functions of various kinds. For example, simulated annealing and tabu search use cost or objective functions. As discussed earlier, however, this confuses the issues of search control and evaluation. One can design a cost function so that it evaluates schedules as final states, but it is much better to design the cost function to take into account the neighborhood of states. The latter approach can help to reduce local maxima that might stall the search. Thus the cost function, and certainly other parameters such as annealing's "cooling schedule" (by which the temperature parameter is adjusted), can profitably encode global search control information.

Unfortunately, we believe that the design of such cost functions and other search parameters can be a significant knowledge-engineering challenge. In DTS, the user is only responsible for providing a "genuine" utility function (i.e., describing preference over output schedules) and identifying key heuristic features of the domain, which the system automatically associates with a direct representation of the relevant search control statistics, learned through experiments with training problems. Furthermore, this search control information is in terms of distributions over attributes of the utility function. Thus the search control information is valid even if the utility function is modified. In our work on scheduling applications, we have found that utility information can be elicited from users in a reliable and straightforward manner.

In addition, traditional uses of objective functions ignore the most important attribute of any objective function—computation time. If computation time were not important, optimization would be a trivial task. It is the constraint of arriving at good solutions in reasonable time that defines the problem, and this tradeoff is explicitly represented in DTS. The explicit use of this tradeoff in selective search is one of the transfers from earlier research on BPS.

4. Conclusion

The use of Bayesian probability theory in DTS underscores that scheduling involves decision-making under uncertainty, and illustrates how imperfect information can be modeled and exploited. The use of multiattribute utility theory in DTS underscores that scheduling involves complex tradeoffs among user preferences. By addressing these issues, the DTS prototype has demonstrated promising performance in preliminary empirical testing. The authors are currently developing a commercial version of DTS at Heuristicrats Research, Inc. (available in 1994).

Bibliography

Cox, R. T. "Probability, Frequency and Reasonable Expectation." *American Journal of Physics*, vol. 14, 1946.

Dechter, R. and J. Pearl. "Network-Based Heuristics for Constraint-Satisfaction Problems." In *Search in Artificial Intelligence*, L. Kanal and V. Kumar, eds., Springer-Verlag, New York, 1988.

de Finetti, B. *Theory of Probability.* John Wiley, New York, 1974.

Duda, R. O. and P. E. Hart. *Pattern Classification and Scene Analysis.* John Wiley, New York, 1973.

Ettinger, G. J. and T. S. Levitt. "An Intelligent Tactical Target Screener." In *Proc. of the 1989 DARPA Image Understanding Workshop.* Morgan Kaufmann, San Mateo, CA, 1989.

EUVE Guest Observer Center. *EUVE Guest Observer Program Handbook.* Appendix G of NASA NRA 92-OSSA-5. Center for EUV Astrophysics, Berkeley, January 1992.

Fox, M. S. *Constraint Directed Search: A Case Study in Job-Shop Scheduling.* Pitman, London, 1987.

Hansson, O. and A. Mayer. "Heuristic Search as Evidential Reasoning." In *Proc. of the the Fifth Workshop on Uncertainty in Artificial Intelligence*, Windsor, Ontario, August 1989.

Hansson, O., A. Mayer, and S. J. Russell. "Decision-Theoretic Planning in BPS." In *Proc. of AAAI Spring Symp. on Planning*, Stanford, 1990.

Hansson, O. and A. Mayer. "A New and Improved Product Rule." Talk presented to *The Eighth International Congress of Cybernetics & Systems*, New York, June 1990[a].

Hansson, O. and A. Mayer. "Probabilistic Heuristic Estimates." *Annals of Mathematics and Artificial Intelligence*, 2:209--220, 1990[b].

Hansson, O. and A. Mayer. "Decision-Theoretic Control of Artificial Intelligence Scheduling Systems." HRI Technical Report No. 90-1/ 06.04/5810, Berkeley, CA, September 1991.

Hansson, O. *Bayesian Problem-Solving applied to Scheduling*. Ph.D. Dissertation, Univ. of California, 1994.

Heckerman, D. E. *Probabilistic Similarity Networks*. MIT Press, Cambridge, 1991.

Howard, R. A. "Information Value Theory." *IEEE Transactions on Systems, Man, and Cybernetics*, vol. SSC-2, 1965.

Johnston, M. D. "Knowledge-Based Telescope Scheduling." In *Knowledge-Based Systems in Astronomy*, A. Heck and F. Murtagh, eds., Springer-Verlag, New York, 1989.

Keeney, R. L. and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Tradeoffs*. John Wiley, New York, 1976.

Lee, K.-F. and S. Mahajan. "A Pattern Classification Approach to Evaluation Function Learning." *Artificial Intelligence*, vol. 36, 1988.

Mayer, A. *Rational Search*. Ph.D. Dissertation, Univ. of California, 1994.

Pearl, J. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, San Mateo, CA, 1988.

Raiffa, H. and R. Schlaifer. *Applied Statistical Decision Theory*. Harvard University, 1961.

Russell, S. J. and E. Wefald. "Principles of Metareasoning." In *Proc. of the 1st Conference on Principles of Knowledge Representation and Reasoning*. Toronto, 1989.

Savage, L. J. *The Foundations of Statistics*. Dover, New York, 1972.

van Hentenryck, P. *Constraint-Satisfaction in Logic Programming*. MIT Press, Cambridge, MA, 1989.

von Neumann, J. and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

von Winterfeldt, D. and W. Edwards. *Decision Analysis and Behavioral Research*. Cambridge University Press, 1986.

Zweben, M., M. Deale and R. Gargan. "Anytime Rescheduling." In *Proc. of the DARPA Planning Workshop*, Morgan Kaufmann, San Mateo, CA, 1990.

Guided Forward Search in Tardiness Scheduling of Large One Machine Problems

Thomas E. Morton[1] and Prasad Ramnath[2]

[1]Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213, tmorton@cmu.edu

[2]Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213, bandit@cmu.edu

Abstract

Giant telescopes need a very high quality pilot telescope to guide them. Similarly, we show here that advanced heuristics for large one machine problems for the weighted tardiness objective also benefit from outstanding guide heuristics. These advanced heuristics can be categorized as having middle or high computational requirements. In other applications, forward algorithms and planning horizon procedures have reduced problems that are high-order polynomial to a complexity that is linear or quadratic in the time horizon; we achieve this here as well for these advanced procedures. We first develop X-dispatch bottleneck dynamics heuristics that allow for inserted idleness which we use as guide heuristics (low computation); next we develop forward algorithm neighborhood search (middle computation) and tabu search methods (high computation). Extensive testing and comparisons with other heuristics are reported. Extensions to job shops and other objectives are planned.

keywords: guide heuristics, planning horizons, scheduling and tabu search.

1. INTRODUCTION

1.1 Overview

High-resolution telescopes need a pilot telescope to guide their search to one tiny part of the sky. The pilot telescope must be of the same exacting quality as the main telescope. We use for our pilot telescope several improved versions of bottleneck dynamics, broadening "dispatch" procedures to "*X-dispatch*" procedures to handle "hot jobs" (by allowing for inserted idleness), and serve as *guide heuristics* by providing starting solutions for more advanced methods like tabu search or neighborhood search. These heuristics are named *X-RM*, *KZRM* and *X-KZ*.

Forward algorithms and planning horizon procedures take advantage of the fact that information further and further into the future is less and less important in changing the optimal first decision. In many cases of dynamic problems in inventory, production smoothing, and similar areas, forward procedures have reduced problems that are NP-complete (or high order polynomial complexity) to a complexity of O(hT), where h is the rolling horizon chosen, and T is the time length of the problem to be solved. (Here we do not use O(hT) in the precise worst-case sense, but in some average-case sense.) We demonstrate here that heuristics such as tabu search can achieve high accuracy for very large problems in a similar fashion with (rough) computational complexity $O(T^2)$. We exploit the fact that the neighborhood in which an interchange needs to be made (while conducting either neighborhood search or tabu search) is limited, either because the initial solution generated by a good guide heuristic is not far away from the optimal, or because the dynamic nature of large problems limits the distance from which a sequence can be placed away from its true optimal sequence. We call this limited-neighborhood search procedure the *k-move*, where "k" is the length of the horizon. In addition planning horizons arise from the fact that both dynamic arrivals and inserted idleness limit busy periods in any job schedule. If an interchange does not disturb the subsequence within a busy period to cross over into the next busy period, then again a planning horizon arises, which means that when computing the cost and effect of the new schedule we needs to limit our attention only to the local effects in the subsequence that was changed. Static problems or very long busy periods allow for a planning horizon effect as well; because computing the changes in the subsequence is limited to the changes in the jobs that formed the neighborhood. We develop a procedure which we call *forward tabu search* by embedding these planning horizon procedures in classical tabu search.

We may distinguish three useful regions of (computational cost, accuracy) for scheduling heuristics:

Type I. low cost	--- 1 simulation of problem
Type II moderate cost	--- 5 to 20 simulations
Type III high cost	--- 100 to 10,000 simulations

Often Type I is used as a guide for Type II, and Type II as a guide for Type III. A good Type I heuristic might be a single-pass myopic dispatch heuristic or the X-dispatch procedures developed in this paper for the one-machine problem. Good Type II heuristics are the shifting bottleneck algorithm (SBI), neighborhood search methods and the multi-pass methods (using leadtime and/or price iteration) using bottleneck dynamics dispatch heuristics. A good Type III heuristic would be forward tabu search or simulated annealing.

1.2 Summary of the Paper

In Section 2 we discuss prior research. Section 2.1 summarizes Type I, Type II, and Type III heuristics for the job shop for various objective functions and supporting computational studies. In Section 2.2 we summarize previous work on X-dispatch procedures and on planning horizon methods, as well as work on tabu search. Section 3 discusses the development of the guide heuristics and forward tabu search methods. Section 3.1 develops three new Type I guide heuristics for the objective function of weighted tardiness. These methods serve as new scheduling heuristics in their own right. Section 3.2 develops forward tabu search and the planning horizon procedures embedded in them. Section 4 presents computational studies which have been performed to date for the one-machine problem. In Section 4.1 we discuss the general design of the study while in Sections 4.2 we present the test results for the Type I guide heuristics (Phase I). The Type II methods discussed in Section 4.3 use neighborhood search to improve the Phase I solution to a local minimum (Phase II), while using Type III, we extend these results using forward tabu search thus further improving upon the neighborhood results (Phase III). Finally, in Section 5 we discuss some of the conclusions and extensions of this article.

2. RELATED RESEARCH

2.1 Type I, Type II, Type III Heuristics

Usually, one-pass myopic dispatch heuristics use as the priority rule that rule that is optimal (or nearly optimal) when considered locally. [Conway, 1965] shows that the myopic rule for minimizing average lateness (namely, the SPT rule) is very robust[1] for the job shop when compared to a large number of other Type I rules. The robustness of myopic rules for Type I heuristics has also recently been demonstrated in a large study for several objective functions and a number of Type I rules by [Lawrence and Morton, WIP].

The dispatch procedures for the job shop schedule one machine at a time, reducing the problem to a one-machine scheduling problem at any decision epoch. One problem with the one-pass myopic dispatch

[1]By robustness, we mean that the accuracy of the solution is not very sensitive to the parameters of either the problem or the heuristic. In the case of the SPT method, this would mean a lack of sensitiveness to the problem parameters.

heuristics for scheduling job shops is that for objective functions involving due dates, a "local" , "operation" or "derived" due date must be estimated for the current machine, thus requiring a good estimate of the leadtime from the current machine under consideration (for scheduling) until job completion. One way to produce a good Type II heuristic requiring on average about five to ten simulations per policy, is a multi-pass myopic dispatch with lead-time iteration where initial estimates of leadtimes are refined for each iteration using information obtained in the previous one. Lead-time iteration has now been shown to be robust in a number of studies, such as [Vepsalainen and Morton, 1987] and [Morton and Pentico, WIP]. The recent study mentioned above [Lawrence and Morton, WIP] verifies that myopic dispatch rules refined by leadtime iteration are consistently superior to the myopic dispatch method alone. Since due dates are well known for the one-machine problem, multi-pass dispatch procedures are usually not applicable in this domain, thus leading to the dichotomy of having Type I dispatch procedures for single machine scheduling problems, and both Type I and Type II dispatch procedures for the more general job shop scheduling problem.

While iterated myopic dispatch methods are excellent methods when there are no congested resources, a number of "bottleneck" Type II methods have proved more effective when certain resources are more critical than others. Such methods include OPT, the shifting bottleneck algorithm, and bottleneck dynamics.

While the details of OPT are proprietary, we know that it schedules as if the limited available processing time on the bottleneck machine were to be conserved rather than that of the current machine as would be for myopic methods. Using these principles in a study on the weighted flow flowshop [Morton and Pentico, WIP] show that OPT-like methods are more effective when there is a single strong bottleneck downstream, but that the myopic method is remarkably robust overall.

The shifting bottleneck algorithm [Adams, Balas, and Zawack, 1988] (SBI version) augments the OPT method (for the makespan criterion) by trying differing machines and then combinations of the machines as the stationary bottleneck. SBI has proven to be fast and accurate, for medium-sized job shops. It uses several simplifications specific for the makespan objective. It uses a NP-complete sub-routine that however is fast in practice (for the relatively small problem sets that were studied). Thus, it is not clear whether or not these ideas can be extended easily to other objectives or to really large job-shop problems.

Bottleneck dynamics is a hybrid between the myopic methods and OPT. It allows a combination of machines to be "critical", and allows the appropriate combination weights (machine "prices") to shift dynamically as the problem progresses. (The "price" is the current opportunity cost of using the machine). Different methods for estimating prices is a rich subject in itself which we do not elaborate here. Several studies have shown that the solutions to jobshop scheduling problems are not too sensitive to the methods used for determining these prices [Lawrence and Morton, 1989]. A more recent study across objective functions [Lawrence and Morton, WIP] shows that bottleneck dynamics consistently outperforms myopic leadtime

iteration by a wide margin. The subject of bottleneck dynamics is developed very completely in [Morton and Pentico, 1993].

It is no accident that all Type III studies to date for the job shop problem have been for the makespan objective function (and for small to medium rather than large job shop problems). The special structure, and smaller size, allows much faster computation in several ways:

1) smaller one-machine subproblems are solved quickly by Carlier's algorithm (which is NP-complete) [Carlier, 1982];
2) a given policy can be evaluated by solving a longest path problem;
3) bottleneck machines and jobs are more sharply defined.

[Adams, Balas, Zawack, 1988] ran SBI (Type II heuristic) and SBII (Type III heuristic) on a number of job shop problems, and compared the results with some Type I heuristics, and with some rather ad hoc Type II heuristics. We will not report these comparisons here; for brevity we restrict ourselves to numerical cross-comparisons with other Type III methods.

Roughly, SBI treats one or more machines as bottlenecks, and solves one machine problems repeatedly, searching over possible bottlenecks. In some easier cases, optimality can be verified. SBII then improves these results by a search method akin to beam search. (thus, SBI is the guide heuristic for SBII).

[Van Laarhoven, Arts, and Lenstra, 1992] have provided both theoretical results and testing for similar job shop problems using simulated annealing :

1. They give proofs of mathematical conditions under which simulated annealing converges asymptotically in probability to an optimum.
2. They give methods for makespan dramatically reducing the part of the neighborhood which must be searched.

Simulated annealing starts with an initial solution obtained from some Type I method and then perturbs this solution by choosing the next move probabilistically. This procedure is repeated continuously with the probability of making a certain move falling off exponentially with the deviation of the solution to that move from that of a deterministically estimated "good" move. The rate of fall-off is controlled by the "temperature": a low temperature gives a result similar to neighborhood search (only a move that improves upon the current solution is chosen as the next); a high temperature, a much more random result. A rough result was that simulated annealing at low temperatures can significantly improve SBII but at about five times the computational cost.

[Dell'Amico and Trubian, 1992] have provided a tabu search method which is perhaps the definitive method to date for the medium-sized job shop problem for the makespan objective. Some features of their method include:

Guide Heuristic--Alternate scheduling forwards and backwards using sort of a bi-dispatch method. (Type I)

Random Starting Points--Always starts at initial heuristic, randomization comes entirely from tie-breakers inside the routine.

Filtering Neighborhoods and Interchanges--Use the excellent makespan properties for simulated annealing makespan results derived by [Van Laarhoven, Arts, and Lenstra, 1992].

Genetic algorithms are an extremely generic class, which can be conceived to contain the other methods as special cases. Good solutions can be allowed to "mate" either bisexually or unisexually. (One unisexual method could be pairwise interchange of priorities of the solution, for example). New children solutions are produced in the neighborhood of the old. A certain number of the old solutions that are bad are allowed to "die" to accommodate the new children. The results will be discussed below. This method could be sharpened dramatically by using the specific problem insights from the other methods.

[Della Croce, Tadei, Volta, 1992] give a comparison of accuracy and running time for all these methods for 10 problems from the set initially considered by Adams, Balas and Zawack (ABZ set). We present this comparison here:

Comparing Methods on Makespan Problems

	Normalized Time	Rel Error	Figure of Merit
Genetic Algorithms	100	1.4	0.01
SBII	1	1.1	0.90
Simulated Annealing	5	0.4	0.50
Tabu Search	1	0.2	5.00

The third column denotes a "Figure of Merit" computed here as 1.0/(Time * Error). While this measure is very crude, it shows Tabu Search as clearly the best method.

2.2 Previous X-Dispatch, Planning Horizon, and Tabu Search Results

X-Dispatch procedures are dispatch rules that allow for inserted idleness. Very little has been done previously to produce Type I or Type II heuristics, which allow inserted idleness into the schedule, except for the makespan objective. The following proposition is known, and is useful:

Proposition: For any regular objective, no operation may be considered to be scheduled next on a given machine, unless it is currently available in the queue, or else will arrive before the current time plus the processing time of the shortest job in the queue.

Proof: Suppose to the contrary. Then one could schedule the shortest available job in the gap, leading to a contradiction.

Definition: Let the jobs that can be scheduled using the above Theorem be termed the extended queue.

Corollary: (by Jack Kanet) For any regular objective, no operation may be considered to be scheduled next on a given machine, unless it is

currently available in the queue, or else will arrive before the minimum finishing time of any job in the extended queue.

Carlier's algorithm [Carlier, 1982] is an exact X-dispatch procedure for the single machine problem for the objective of minimizing makespan. Thus SBI applied in [Adams, Balas, Zawack, 1988] is a X-dispatch heuristic. For small job shops it may be considered to be a Type II method. However, the procedure is NP-complete, so for large job shops it must be treated as Type III (computationally expensive).

The earliest reference on inserted idleness for Type I and Type II heuristics that we are aware of, except for makespan as discussed above, is a suggestion in [Morton and Pentico, 1993, Ch. 7] for a rough modification of bottleneck dynamics priorities for any objective to account for inserted idleness. This actually forms the basis of our X-dispatch procedure for weighted tardiness and similar objectives, and so we defer further discussion of this until Section 3.2. Work currently in progress indicates that these multi-pass X-dispatch procedures do very well in the context of scheduling job shops to minimize weighted tardiness. The modifications to account for the job shops is very similar to other bottleneck dynamics procedures modified from one-machine scheduling problems. The heuristic accounts for local due dates (using leadtime iteration) as well as the time at which hot jobs arrive from upstream machines. Another recent reference to inserted idleness for single machine scheduling is [Zhou and Sridharan, 1993] where they apply inserted idleness to the scheduling of a single machine for the weighted tardiness objective. The algorithm used is a modification of the "KZ" procedure developed in [Kanet and Zhou, 1993].

Planning horizon or rolling horizon procedures have a long history of being applied in practice, pre-dating academic research on them. Basically a producer faces a long imperfect forecast using which periodic control decisions (at frequency t) must be made. (E.g. which job to sequence next.) A planning horizon c and a forecast horizon T (where T is greater than or equal to c are chosen. At time t a finite problem from t to t+T is solved, using the best available forecast. Some simple salvage function may or may not be attached to the further future from t+T (which is unknown or inaccurate) to solve this T-length problem. The solution from t to (t+c) is implemented. Then at time (t+c) a new T-length problem from t+c to t+c+T using improved forecasts which have subsequently become available, and any salvage correction is used. The solution from (t+c) to (t+2c) is then implemented, and so forth. There are two main advantages from using such a rolling horizon procedure:

(a) Typically the value to the accuracy in the first-period decisions from increasing T goes down very rapidly as T increases due to the corrective effects of intervening decisions. Many times the effect after a point can be proven to be 0 (Exact planning horizons), more often it can be proven to be small (Approximate planning horizon).

(b) The accuracy of forecast information almost always dies off rapidly as T is increased, so that the problem needs to be re-solved periodically after updating the forecast.

A full consideration of the determination of the effect of horizon length on solution quality leads into the subject of forward algorithms and planning horizons. Here one solves a longer and longer version of the problem in an efficient forward manner, and determines by some theoretical or practical matter as to what stopping value of T one may use to safely stop the algorithm.

It is not our purpose here to re-develop the ideas of rolling horizons, forward algorithms, and planning horizons in detail, or to mention in detail the large literature. We are only trying to point out that there is wide experience in many different categories of problems (deterministic or stochastic, convex or integer, one variable or many), where rolling horizon procedures using forward algorithms with relatively short horizons can accurately solve problems with much longer horizons. We use this insight to design heuristics, but then use empirical methods to verify the concept and to determine the length of the horizons.

A good survey on this subject is [Morton, 1981]. Early classic articles include [Arrow and Karlin, 1958], [Cyert and March, 1963], [Johnson, 1957], [Manne and Veinott, 1967], [Modigliani and Cohen, 1961], [Modigliani and Hohn, 1955]. A good discussion of horizon length for Markov decision processes is given in [Morton and Wecker, 1977]. An illustration of planning horizons in stochastic problems is given in [Morton, 1978] while good theoretical foundations are given in [Morton, 1979].

The foundation work in tabu search is due to Fred Glover. A good primer in the area is his survey article [Glover, 1990]. Another recent paper is [Glover and Laguna, 1989].

3. GUIDE/FORWARD HEURISTICS

In Phase I, improved Type I guide heuristics are developed for the one machine problem. In Section 3.1 three new guide heuristics, "X-RM", "KZRM" and "X-KZ" are developed for the one-machine dynamic weighted-tardiness problem. In Section 3.2 forward tabu search is developed for the weighted tardiness problem. Here, we also discuss how planning horizons can be embedded within search procedures for scheduling problems.

3.1 Improved Phase I (Guide) Procedures for Weighted Tardiness

While the R&M procedure [Rachamadugu and Morton, 1982] has been tested well in a number of situations as a relatively simple heuristic of medium accuracy for the objective function of minimizing weighted tardiness, it is worth considering a number of ways to sharpen it before considering it as a high quality guide heuristic for forward tabu search and/or for other Phase III procedures. We consider three types of improvement.

1. An inexpensive approximation to beam search with the beamwidth = 1, and a low-cost horizon approximation to the probe.[2] Here we consider a combination of the recent K&Z [Kanet and Zhou, 1993] approximation with an R&M probe, which we term KZRM.

2. Sharpening of weighted tardiness heuristics to allow inserted idleness by applying a correction to the R&M priorities. We call the resulting heuristic X-RM.

3. Sharpening of KZRM to allow inserted idleness by adding hot jobs to the queue jobs already being considered. This approach requires an estimate of the machine price R(t). Two variants are tried. One prices the machine on the basis of the current queue, and one on a more sophisticated estimate of the current busy period.

K&Z with R&M Probe (KZRM). This is a dispatch procedure. Whenever a job must be chosen, calculate the current R&M priorities for every job in the queue. Suppose L jobs are currently in the line. Calculate the estimated cost of L scenarios, where each scenario in turn has one of the L jobs as being scheduled first, and the remaining jobs as being scheduled after the first by their current R&M priorities. Note that the L completion times (from these L jobs), enable the calculation of the objective function cost for each job and can be added to obtain the total cost of each scenario. This gives us a valuation of each scenario. Now evaluate each of the L scenarios, and choose the scenario with the lowest cost. The correct scheduling choice is then the first job as found in scenario with the lowest cost. We move the simulation to the completion time of this job and repeat the procedure again for the L-1 remaining jobs.

Inserted Idleness by Priority Correction--X-RM. The bottleneck-dynamics procedure for the weighted tardiness one-machine dynamic problem is quite adequate as dispatch procedures go [Morton and Pentico, Ch. 7, 1993]. However, it has a serious defect when it comes to very high accuracy procedures, in that no inserted idleness is allowed for late-arriving "hot jobs". Since it is known that the only jobs which need to be considered to be scheduled next at time t are those in the queue, or those which will arrive before $t+p_{min}$ (Current time + Shortest processing time among all jobs currently in the queue, from the proposition in 2.2), a natural hope would be to reduce scientifically the priorities of such soon-to-arrive hot jobs in proportion to the inserted idleness that they could impose on the machine. [Morton and Pentico, Ch. 7, 1993] suggest a simplified procedure to do this:

Calculate priorities as in a dispatch procedure, except that jobs soon to arrive are included at a reduced priority:

$$\pi_j' = \pi_j [\, 1.0 - B(r_j - t)^+ / p_{min}]$$

---

[2]The probe is the evaluation procedure used in beam search to evaluate each candidate node, before picking the best "b", "b" here being equal to the beamwidth.

where π_j is the current priority of the job (in the case of X-RM, this being the priority generated by the R&M rule), B a constant, t the current time, p_{min} the shortest processing time among all jobs in the current queue, and r_j the time the job arrives at the queue. In words: "The original priority of a not-yet-arrived job is degraded by a term proportional to the idleness involved as a fraction of the waiting job with the smallest processing time."

The proportionality was found to increase linearly with the utilization factor. One of the authors in a preliminary set of experiments fit B as $B = 1.3 + \rho$ where ρ is the utilization factor. The same preliminary experiments suggest that the X-dispatch procedure produced by reducing hot job priorities in this fashion, and then scheduling according to the dispatch procedure produces an improvement of 40% to 45% in the ordinary R&M dispatch procedure for weighted tardiness.

Inserted Idleness by Approximate Beam Search X-KZ. Def ine the extended queue as the line at time t plus all jobs arriving before time $t+p_{min}$. Consider a scenario after putting each job in the extended queue. Evaluate each scenario as in KZRM, except that for the scenarios starting with idle time, add a cost equal to the machine price times the extra completion time of the scenario due to the inserted idle time. There are two variants, depending on whether the machine price is estimated from the line length, or an estimated busy period. In preliminary experiments, the line length method turned out to be relatively robust and easier to use, and therefore was adopted.

3.2 Forward Tabu/Neighborhood Search

Here we do not use "tabu search" as synonymous with "using a local move control mechanism" but rather as synonymous with "extended neighborhood search", that is, a neighborhood search method with some controls for diversification, intensification, and stopping. Of course, this leaves us with a rather large number of design parameters. We shall discuss those which seem of most interest here:

1. Neighborhood design.
2. Forward approximate interchange/move.
3. Tabu list design.

Neighborhood Search. Until recently the main neighborhoods which researchers considered for an n-job sequencing problem were: adjacent pairwise interchange requiring about n, and general pairwise interchange requiring about $n^2/2$ interchanges. It has been realized that adjacent interchange is not very accurate, but also that general pairwise interchange for large problems is very expensive.

We have crafted an effective and more usable procedure called "k-move". The idea is to choose an element, and move it j moves to the left or right, where $j <= k$, where k is fixed in advance. The k-move procedure has about (2k-1)n required evaluations (ignoring boundary effects), so that the cost of the procedure can be accurately controlled.

Furthermore, only small values of k are usually cost-effective.[3] For example we recently ran Phase I for 100- and 200-job problems for the objective of minimizing total weighted tardiness for single machines. Adjacent pairwise interchange reduced Phase I errors by a factor of 3 or 4; 5-move reduced them again by a factor or 4 to 5; 20-move gave only modest additional gains. The idea of augmenting general pairwise interchange with all k-moves up to and including n-moves is not new. Ignoring interchange moves entirely, and limiting k to 5 or 20 seems to be an effective new procedure.

In addition, specific problems can be filtered in various ways. For makespan problems, for example, no job should be moved earlier unless it is currently on the critical path. The critical path, in turn is limited to a single busy period. Similarly, for weighted tardiness, no job should be moved earlier unless it is currently tardy.

When Phase I heuristics provide priorities for jobs as part of their output, one can also filter by setting an error range E, and not moving a job with priority P earlier than a job with priority Q>P+E, for example. This approach however, has not been implemented in our computational study.

Forward Approximate Interchange/k-Move. The usual method for evaluating an interchange or k-move for a large one-machine problem would be to simulate the entire problem again for the new sequence after an interchange/move. This would involve a simulation from the first changed-sequence job at time t all the way out to T, which requires a time O(n), where n is the number of jobs. Instead, following a forward algorithm/planning horizon procedure, we simulate out to some time t+h where h is determined as the period after t at which the modified sequence and the original sequence are identical. That is h represents a planning horizon in evaluating the change in valuation of the sequence. Planning horizons arise in a number of ways. First, busy periods always produce planning horizons if the idle period is fairly long. A change resulting from a move impacts the jobs subsequent to the position of the move by changing their completion times. However, if a period when the queue is empty arises in the sequence at t+h, then, this change does not get propagated any further, unless the changes resulting from the move result in the new completion times reducing this period of idleness to zero. Second, inserted idleness produces planning horizons as well (a benefit in addition to the improved solution) by allowing the impact of the new schedule to be limited to the period t+h when the inserted idleness occurs in the solution (in both the new and the old

---

[3] A move only displaces one job, namely, the one considered as being moved away from its current position. If the procedure used to generate the initial solution (in Phase I) is appropriate, then, it would be reasonable to assume that jobs are displaced not far away from their optimal positions in their schedules, hence the limitation of the distance of the move to some "k". Furthermore, interchanges assume that pairs of jobs are displaced equally, whereas a move such as in a k-move assumes that each job is displaced in a different manner than the other jobs.

ones). Third, if the shop is extremely busy, then the jobs that are impacted by the move are only those within a distance of k from the job being moved. This is similar to moves being implemented in static problems, where jobs outside the neighborhood do not incur any change in their completion times. In this last case the horizon, h, is in fact, equal to k. To avoid the chance that h might be large, we also have some cutoff value t+H and require h<= H. H might be a pre-determined constant, or it might represent the end of the current busy period, or of the next busy periods. In each of these instances the computations for the new schedule would involve finding the changes from the current job onwards till the end of the horizon, since the rest of the schedule remains unchanged. This makes computing the objective function and completion times extremely fast.

The similar analysis for a multi-machine shop involves one sequence for each machine, but H is total time in the simulation, not for any one machine. Otherwise little is changed. Simultaneous ends of busy periods will not be common, therefore exact planning horizons will be less frequent. However, the busy period on the highest price machine is likely to act as a near horizon.

Tabu List Design. We prefer simplicity in tabu list design. We either keep no list at all (ordinary neighborhood search), or an "infinite" list, recording every previous position completely. Thus the rule is either "no restrictions, stop at local minimum", or "never repeat a move". This rule comes from the fact that for very large problems, computation time for k-moves is much larger than the log-linear search time to check for duplicated moves. We have also consistently found that longer lists gives better results and do not get trapped into solutions where all moves are tabu. Due to the mechanics of keeping efficient networks to access the list, it is also actually cheaper in overhead, to keep the full list than a partial one.[4]

Notice that in deciding whether a move is tabu, we do not have to compare the full solution structure. We may access just the objective value, and go to the full structure only for tie-breaking.

---

[4]The tabu list is stored as a binary tree based on the objective value of each sequence. To search for a certain move as being tabu, the tree is searched until the nodes in it have the same solution as that from the potentially new move. If there is a match, these nodes alone are examined to verify whether they have the same schedule as that in the potential move. Since the search time is log-linear in a binary tree, this seems to be the most efficient way to store tabu lists. However, if the tabu list were in fact finite, then the binary tree will have to be updated at certain fixed intervals, by truncating from it all the nodes older than a certain age. This seems more efficient than truncating the oldest node after each move. Therefore for finite lists, the tabu list oscillates between two sizes, the smallest size s* after truncating the set of oldest nodes, and s** just before truncation (which means s**-s* nodes are truncated). In practice, however, we observe that this truncation procedure involves recreating the binary tree from scratch thus making the computational overhead more than by just keeping the entire tabu list without any truncation.

4. COMPUTATIONAL STUDIES

4.1 Problem Design

The computational study in this section consists of testing the Type I, Type II and Type III heuristics, for the one machine problem to minimize total weighted tardiness. A total of n jobs arrives dynamically. Without loss of generality the mean processing time p_{av} is taken as 100. The standard deviation is set at 50 (compare negative exponential which would give 100), (the smallest processing time is truncated at 1). Weights have the same distribution as processing times. The utilization factor ρ controls shop loading at (ρ = 0.6, 0.75, 0.9). The Poisson arrival rate λ for forecast arrivals is controlled by Little's Law $\lambda = \rho/p_{av}$. Due dates are set a fixed multiple κ times processing time after forecast arrivals: $d_j = f_j + \kappa p_j$. We set κ experimentally to achieve certain tardiness factors τ representing the proportion of jobs tardy for a good heuristic (τ= 0.1, 0.4, 0.7) (That is, τ is determined experimentally as a function $\tau(\rho,\kappa)$, and the correct κ in each cell for a given τ and ρ is estimated by interpolation.) Finally, the actual arrival times to the machine r_j are randomly distributed from the forecast f_j, by $r_j = f_j + \varepsilon_j p_j$, so that finally $d_j = r_j + (\kappa + \varepsilon_j)p_j$. Here ε_j has mean 0 and standard deviation s_ε (s_ε = 1, 2, 3).

(Note: in the actual study the randomness occurred at the due date itself rather than in the difference between forecast and actual arrival time. We did some subsidiary testing to establish that this difference from the model had an insignificant effect.)

The central parameter set was ρ=0.75, τ=0.4 and s_ε = 2. The other parameter sets involved varying one parameter at a time from these central values. That is, two runs were experimented with τ at 0.1 or 0.9 while the others were at the central; two runs were experimented with ρ at 0.6 or 0.9, and two runs experimented with s_ε at 1 or 3, giving a total of seven parameter sets.

Phase I Study. The heuristics investigated for weighted tardiness included:

RAND	Random
FCFS	First Come First Served
CRIT	Critical Ratio
SLACK	Least Slack
EDD	Earliest Due Date (Operation Due Date)
MODD	Modified Operation Due Date
WSPT	Weighted Shortest Process Time
WCOVERT	Weighted COVERT (k=2)
KZ	Kanet and Zhou
KZRM	Kanet and Zhou modified by Rachamadugu and Morton
RM	Rachamadugu and Morton

X-RM	Inserted Idleness Priority Correction for RM
X-KZRM	Inserted Idleness Modification of KZRM

Definitions of these heuristics are given in Appendix A. For each heuristic, and each parameter variation, a n=5000 job problem was solved and replicated 16 times. Thus for the objective of minimizing total weighted tardiness a total of 13 x 7 x 16 = 1,456 separate 5000 job problems were solved, involving a total of about 7 million simulated jobs.

4.2 Phase I Testing for Weighted Tardiness

TABLE 1. Phase I Weighted Tardiness Study
(Normalized Heuristic Costs)

	Base#	τ=.1	τ=.7	ρ=.6	ρ=.9	s_ε=1	s_ε=3
RANDOM	.999	.999	.999	.999	.999	..999	.999
FCFS	.797	.289	.954	.862	.832	.717	.815
CRIT	.703	.312	.800	.761	.729	.650	.715
SLACK	.653	.198	.916	.697	.791	.680	.566
EDD	.614	.188	.891	.645	.781	.644	.532
MODD	.336	.116	.454	.394	.404	.349	.305
WSPT	.186	.298	.136	.296	.087	.148	.242
WCOVERT	.065	.107	.138	.183	.029	.056	.092
KZ	.067	.035	.131	.188	.046	.051	.095
KZRM	.054	.017	.103	.166	.035	.028	.084
RM*	.038	.001	.097	.173	.002	.028	.061
X-RM	.008	.002	.031	.042	.001	.019	.004
X-KZ	.002	.014	.003	.002	.027	.001	.007

#Base is τ=0.4, ρ=0.75, s_ε=2

*Parameter set at k=2

Table 1 summarizes the results of Phase I for weighted tardiness. The performance measure in any cell is the average of 16 separate 5000 job runs. The performance measure is relative rather than absolute: $P = (C_{heur} - C_{best})/(C_{worst} - C_{best})$. Here C_{worst} is the worst result (usually from RANDOM) , and C_{best} is the best solution among all the heuristics for each problem. Therefore, each cell in the table represents the average P for 16 problems of that particular type.

4.3 Improving Phase I Results to a Local Minimum

Phase I results will not usually be at a local optimum; that is, small changes to the policy from a Type I method may yield an improvement. Thus in Phase II we followed Phase I by neighborhood search (Type II) to find a local minimum.

Phase II Neighborhood Search. We ran the reduced heuristic set FCFS, EDD, WSPT, KZ, RM, X-RM, and X-KZ. For n=200 we ran the 7 parameters sets using 5 replications in each, or 35 runs per heuristic. For n=1000 we ran 7 parameter sets with 1 replication or 7 runs per heuristic. For n=2000 we ran 1 parameter set (the central case) with 2 replications, that is, 2 runs per heuristic. For each run we started from the Phase I result and performed neighborhood k-move search with k=1 (adjacent pairwise interchange until no further improvement was possible; then we switched to k=5, and finally to k=20. See Table 2.

Note that making interchanges for k=1 usually reduces errors by a factor of 3.0 to 3.5; next interchanges for k=5 again reduces errors by a factor of 2.0 to 2.5; finally interchanges for k=20 reduces errors again by 30 to 50%. The case n=2000 does not fit this pattern very well, but since the total sample size was only 2 for this case, little can be inferred.

Note from Table 2 also that the best heuristics from Phase I tend to remain the best after neighborhood search. A major exception is X-KZ which is in first place after Phase I, but is in third place after neighborhood search. The usual order of preference after neighborhood search is X-RM, RM, X-KZ, WSPT, although for the few problems run for n=2000 this order changed somewhat. One possible inference could be that X-KZ, usually the best Type I heuristic, allows for inserted idleness as well, thus possibly locking the Phase I solution in some local minima from which it is difficult to get out of.

TABLE 2. Neighborhood Search Improvement
(Normalized Heuristic Costs)

Heur.	Phase-I	k=1	k=5	k=20
n=200				
FCFS	.946	.253	.0876	.0520
EDD	.745	.226	.0872	.0503
WSPT	.366	.0992	.0207	.0127
KZ	.136	.0643	.0353	.0195
RM	.102	.0341	.0156	.0110
X-RM	.0582	.0189	.0079	.0069
X-KZ	.0430	.0336	.0231	.0136
AV	**.342**	**.104**	**.0396**	**.0237**
n=1000				
FCFS	.999	.322	.158	.109
EDD	.877	.312	.147	.0938
WSPT	.275	.102	.0344	.0189
KZ	.158	.0801	.0545	.0266
RM	.101	.0438	.0237	.0155

X-RM	.0706	.0294	.0168	.0108
X-KZ	.0674	.0584	.0441	.0162
AV	**.364**	**.135**	**.0684**	**.0417**
n=2000				
FCFS	.999	.412	.412	.412
EDD	.860	.320	.282	.282
WSPT	.186	.0571	.0137	.0039
KZ	.0940	.0442	.0246	.0178
RM	.0558	.0191	.0058	.0040
X-RM	.0445	.0149	.0026	.0017
X-KZ	.0374	.0321	.0237	.0197
AV	**.325**	**.128**	**.109**	**.106**

Table 3 shows computation times for these same problems. Note that the best heuristics after neighborhood search also take by far the least total computation time, since these paths started with better solutions from Phase I. Again X-KZ is the exception, it comes in second or third in terms of total computation time, even though it is the most accurate Phase I heuristic.

TABLE 3. Neighborhood Search
Computation Times (Seconds)
(SUN-SPARC Machine)

Heur.	**Phase-1**	**k=1**	**k=5**	**k=20**
n=200				
FCFS	.08	2.63	8.87	17.4
EDD	.08	2.42	7.90	16.2
WSPT	.08	.73	2.66	5.4
KZ	.14	.59	1.96	4.8
RM	.09	.43	1.58	3.3
X-RM	.11	.38	1.19	2.2
X-KZ	.74	.83	1.65	4.2
AV	**.19**	**1.14**	**3.69**	**7.64**
n=1000				
FCFS	.40	78.5	173.	275.
EDD	.41	71.2	170.	274.
WSPT	.40	23.3	90.	170.
KZ	.63	15.9	65.	154.
RM	.43	12.3	56.	135.
X-RM	.51	11.0	38.	92.
X-KZ	2.82	5.7	33.	105.
AV	**.80**	**19.9**	**89.3**	**172.1**
n=2000				
FCFS	.85	534.	1457.	2734.
EDD	.83	447.	1297.	2000.

WSPT	.84	136.	528.	851.
KZ	1.23	82.	325.	688.
RM	.88	64.	251.	399.
X-RM	1.05	66.	248.	298.
X-KZ	4.55	17.	167.	516.
AV	**1.46**	**192.**	**610.**	**1069.**

To check the hypothesis that neighborhood search is of complexity $O(n^2)$, which is to say that the number of iterations required is O(n) (since the time per iteration is known to be O(n)), we would expect running times for n=2000 to average 100 times that for n=200. Table 3 suggests that the factor is actually about 150 instead of 100. Since the planning horizon nature of the interchange process itself suggests that the computation time for an interchange will increase almost exactly linearly in n, it would seem that the number of iterations increases by a factor of 15 for an n increasing by a factor of 10 (which was verified on inspecting the number of iterations before convergence). Thus to a crude approximation, the number of iterations required seems to be $O(n^{1.2})$

Phase III. Full Tabu Search. Next we desired to compare the results of neighborhood search vs. neighborhood search with the addition of forward tabu search (Type III). For each of the seven heuristics, for the central parameter set, we first ran neighborhood search on the Phase I results as in Table 2. That is, first we ran k=1 until no more improvement was possible, then k=5, then k=20. Call this solution the NBHD solution, and the corresponding time T. Now continue running with infinite tabu list tabu search at k=1 until time 2T; continue with infinite tabu list with k=5 until time 4T; continue as before with k = 20 until time 8T. With each new tabu phase, return to the previous best solution before continuing.

We ran 25 problem replications for 50 job problems, 5 problem replications for 200 job problems, and 1 replication for 1000 job problems.

TABLE 4. Forward Tabu Search Vs. Neighborhood Search

HEUR.	PHASE1	NBHD	TABU-1	TABU-5	TABU-20
n = 50	**(25 Replications)**				
FCFS	.9900	.0419	.0405	.0379	.0334
EDD	.8034	.0431	.0413	.0371	.0335
WSPT	.2113	.0138	.0121	.0115	.0098
KZ	.1034	.0154	.0134	.0117	.0110
RM	.0591	.0085	.0076	.0072	.0056
X-RM	.0431	.0072	.0069	.0064	.0063
X-KZ	.0270	.0089	.0087	.0072	.0064
AV.	**.3196**	**.0198**	**.0186**	**.0170**	**.0151**

n = 200	**(5 Replications)**				
FCFS	.9999	.1143	.1131	.0871	.0719
EDD	.7865	.0804	.0795	.0780	.0637
WSPT	.3371	.0346	.0335	.0313	.0290
KZ	.1864	.0417	.0417	.0407	.0305
RM	.1465	.0286	.0286	.0276	.0266
X-RM	.0668	.0072	.0072	.0063	.0029
X-KZ	.078	.0299	.0291	.0277	.0226
AV.	**.3716**	**.0481**	**.0475**	**.0427**	**.0353**

n = 1000	**(1 Replication)**				
FCFS	.9999	.0653	.0652	.0646	.0641
EDD	.8467	.0729	.0729	.0729	.0715
WSPT	.3038	.0173	.0171	.0161	.0161
KZ	.1499	.0212	.0212	.0212	.0207
RM	.0990	.0013	.0012	.0012	.0008
X-RM	.0725	.0017	.0017	.0000	.0000
X-KZ	.0784	.0251	.0251	.0251	.0251
AV.	**.3502**	**.0291**	**.0290**	**.0286**	**.0259**

Looking at the results, it seems reasonable to say that 50 job problems are easier than 200 or 1000 job problems. Average errors starts lower, and are lower at all stages. Neighborhood search reduces errors by a factor of 16, while requiring only 1/8 of the total time. Tabu search reduces errors by about 25% while requiring 7/8 of the total time. For 200 jobs problems, neighborhood search reduces errors by a factor of 7, and tabu search by another 25%. Finally for the single 1000 job problem, neighborhood search produces a factor of 12, while tabu search saves another 11%. (This final sample size is far too small to draw any conclusions, however.)

While the performance of tabu search here may not seem particularly impressive, there may very well be situations where the extra accuracy is worth a high cost. Perhaps a better question is whether or not competitive methods require more or less computation.

5. SUMMARY AND CONCLUSIONS

Improved Type I guide heuristics were developed for large weighted tardiness problems: KZRM, X-RM, and X-KZ. KZRM is an improvement of the recently [Kanet, Zhou, 1993] heuristic, both being inexpensive approximations to 1-beam search. X-RM and X-KZ are improvements to RM and KZRM to allow inserted idleness.

Forward tabu search has been also developed here by embedding planning horizons within them. Computation is greatly reduced for large problems by using planning horizon procedures. For neighborhood search versions of tabu search a new version of interchange called k-move has been created, which is precise and controllable as to accuracy and time used. For tabu lists we have invented efficient storage and

access methods, so that one may usually not maintain a list at all, or make every previous move tabu (infinite tabu list).

In a large initial Phase I study we investigated 13 Type I heuristics on a number of parameters, each cell representing sixteen 5000 job problems. The three best performers for most parameters, and overall were R&M , X-RM and X-KZ. Next in Phase II we improved the Phase I results by neighborhood search, a Type II procedure. Neighborhood search reduces errors by 9 to 16 for three substudies with n=200, n=1000, n=2000. In general heuristics with the largest errors in Phase I retained the largest errors in Phase II, an exception being X-KZRM which was the best performer in Phase I but came in third in Phase II. Computational times for neighborhood search shows that the best Type I heuristics usually also show the least computation time during Phase II. Finally we added forward tabu search to improve these results. The experience here was that while neighborhood search reduced errors by a large factor. additional tabu search up to 8 times the total computational cost typically reduced remaining costs about 25%.

We expect the results to be same if the methods are extended to job shop scheduling. Currently work is under way to investigate these extensions.

BIBLIOGRAPHY

Adams J., Balas E. and D. Zawack (1988) "The shifting bottleneck procedure for job shop scheduling," **Management Science** 34, 391-401.

Arrow K. and S. Karlin (1958) "Production over time at increasing marginal cost," Studies in the Mathematical Theory of Inventory and Production. Stanford CA: Stanford University Press.

Carlier, J. (1982) "The One Machine Sequencing Problem," **European Journal of Operations Research** 11, 42-47.

Cyert R.M., and J.G. March ((1963) A Behavioral Theory of the Firm. Englewood Cliffs, N.J.: Prentice-Hall.

Dell'Amico, M. and Marco Trubian (1991) "Applying Tabu-Search to the Job Shop Scheduling Problem," Politecnico di Milano, Italy

Della Croce, F., R. Tadei and G. Volta (1992) "A Genetic Algorithm for the Job Shop Problem," D.A.I. Politecnico di Torino, Italy.

Glover, F. (1990) "Tabu Search: A Tutorial," **Interfaces** 20, 74-94.

Glover, F. and M. Laguna (1989), "Target Analysis to Improve a Tabu Search Method for Machine Scheduling," Advanced Knowledge Research Group, US West Advanced Technologies, Boulder Colorado.

Johnson, S. M. (1957) "Sequential Planning Over Time at Minimum Cost," **Management Science** 4, 435-437.

Kanet, J. and Z. Zhou (1993) "A Decision Theory Approach to Priority Dispatching for Job Shop Scheduling,", Production and Operations Management, 1993, forthcoming.

Lawrence, S. and T.E. Morton (1989) "Resource-Constrained Multiproject Scheduling with Tardy Costs: Comparing Myopic, Bottleneck, and Resource Pricing Heuristics," Carnegie Mellon University. To appear in the **European Journal of Operations Research**

Lawrence, S. and T.E. Morton "Job Sequencing Via Multipass Dispatch Methods," Work-in-Progress, GSIA, Carnegie Mellon University

Lundin, R. and T.E. Morton (1975) "Planning Horizons for the Dynamic Lot Size Model: Protective Procedures and Computational Results," **Operations Research** 23, 711-734.

Manne, A. and A.F. Veinott Jr. (1967) "Optimal Plant Size with Arbitrary Increasing Time Paths of Demand," Investments for Capacity Expansion, Size, Location, and Time-Phasing. MIT Press: Cambridge, MA.

Modigliani, F. and K. Cohen (1961) "The Role of Anticipations and Plans in Economic Behavior and Their Use in Economic Analysis and Forecasting," Bureau of Economic and Business Research. University of Illinois, Urbana.

Modigliani, F. and F. Hohn (1955) "Production Planning over Time and the Nature of the Planning Horizon," **Econometrica** 23, 46-66.

Morton, T.E. (1978) "Universal Planning Horizons for Generalized Convex Production Scheduling," **Operations Research** 26, 1046-1058.

Morton, T.E. (1979) "Infinite Horizon Dynamic Programming Models--A Planning Horizon Formulation," **Operations Research** 27,730-742.

Morton, T.E. (1981) "Forward Algorithms for Forward Thinking Managers," **Applications of Management Science, Vol. I**. JAI Press.

Morton, T. and D. Pentico "Comparing Myopic, OPT, and Bottleneck Dynamics Heuristics in a Generalized Flowshop," Work-In-Progress, GSIA, Carnegie Mellon University

Morton, T. and D. Pentico (1993) Heuristic Scheduling Systems: With Applications to Production Systems and Project Management, Wiley Series in Engineering and Technology Management, John Wiley & Sons, New York.

Morton, T. and W. Wecker (1977) "Discounting, Ergodicity, and Convergence of Markov Decision Processes," **Management Science** 23, 890-900.

Muth, J.F. and G.L. Thompson (1963) Industrial Scheduling, Prentice Hall, Englewood Cliffs, N.J.

Ow, P.S. and; T.E. Morton (1988) "Filtered Beam Search in Scheduling," **International Journal of Production Research** 26, 297-307

Van Laarhoven, P.J.M., E.H.L. Aarts and J.K. Lenstra (1992) "Job Shop Scheduling by Simulated Annealing," **Operations Research** 40, 112-129.

Vepsalainen, A. and T.E. Morton (1988) "Improving Local Priority Rules with Global Leadtime Estimates," **Journal of Manufacturing and Operations Management** 1, 102-118.

Zhou, Z and V. Sridharan (1993) "Tactically Delayed Scheduling in a Dynamic Single Machine Shop", Working Paper, Clemson University.

APPENDIX. PHASE 1 HEURISTICS

(Slack = $S_j = d_j-(t+p_j)$)

RANDOM	Priority is random
FIRST COME FIRST SERVED	Priority is lowest r_j
CRITICAL RATIO	Priority is $(w_j/p_j)[1/(1+(S_j/p_j)]$
SLACK	Priority is lowest S_j^+ (Some authors use S_j)
EDD	Priority is lowest d_j
MODIFIED EDD	$d'_j = \max(d_j, t+p_j)$, then apply EDD

WSPT	Priority is highest w_j/p_j
WCOVERT	Priority is $(w_j/p_j)[1 - (S_j)^+/kp_j]^+$ (k commonly same as due date setting)
KZ (Kanet & Zhou)	Cost each possible job as first: 1. cost of that job plus 2. remaining jobs in queue at their mean position 3. ignore other effects
KZRM	As for KZ, except schedule remaining jobs in 2.above by R&M heuristic
RM	Priority is $(w_j/p_j)[\exp\{ -(S_j)^+/kp_{av} \}]$ (k commonly = 2)
X-RM	Priority is RM priority times $(1 - (1.3+r)(r_j-t)^+/p_{min})$ p_{min} shortest processing time in queue
X-KZ	As for KZRM except add any hot jobs to first choices, and cost idletime appropriately

An Overview of Tabu Search Approaches to Production Scheduling Problems

J. WESLEY BARNES[1], MANUEL LAGUNA[2], and FRED GLOVER[3]

[1] *Cullen Trust for Higher Education Endowed Professor in Engineering, Graduate Program in Operations Research and Industrial Engineering, Department of Mechanical Engineering, ETC 5.128D, The University of Texas at Austin, Austin, Texas 78712,* wbarnes@mcl.cc.utexas.edu

[2] *Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, Colorado 80309-0419,* manuel@mayan.colorado.edu

[3] *U S West Chair in Systems Science, Graduate School of Business and Administration, Campus Box 419, University of Colorado at Boulder, Boulder, Colorado 80309-0419,* glover_f@cubldr.colorado.edu

Abstract

During the last five years, tabu search has been shown to be a remarkably effective method in solving difficult problems in the timely and very important area of production scheduling. In this paper, we present an overview of the research that has contributed to that success. We also review the various techniques that have been employed, giving attention to advances that have led to major improvements and suggesting directions for future research.

key words: tabu search and production scheduling.

1. Introduction

Tabu search (TS) was introduced over a decade ago and was first fully described in 1986 (Glover 1977, 1986). More recent introductory level reviews include a "User's Guide" by Glover, Taillard, and de Werra (1991) and a "Primer" by Laguna (1992). In essence, TS is an adaptive technique that hierarchically directs one or more local search procedures in aggressive pursuit of the global optimum while utilizing memory functions to avoid becoming trapped in local optima. The TS framework allows a high degree of freedom for designing solution procedures. Researchers have used this flexibility to explore new strategies that may result in more powerful solution methods.

TS has been shown to be remarkably effective, dominating alternative techniques, in a spectrum of arenas ranging from integer and nonlinear programming to sequencing and production scheduling problems. This success been especially marked in TS applications to production scheduling. The purpose of this paper is to provide an overview of these applications and to provide a synthesis of the methods that led to major improvements in solving production scheduling problems.

This paper is organized in five sections. The second and third sections deal with single and multiple machine production scheduling problems, respectively. In the descriptions in Sections 2 and 3, we pay special attention to the features that characterize each TS procedure. Similarities and differences among these features are discussed in Section 4. In the fifth and final section of the paper, we suggest directions for future research.

2. Single Machine Production Scheduling Problems

This section presents the TS based research that has been applied to three types of single machine production scheduling problems, the traveling salesman problem, single machine problems with linear delay penalties and sequence dependent set up costs, and single machine problems with deadlines, holding costs, and sequence dependencies.

2.1 The Traveling Salesman Problem

One of the major early applications and proving grounds for TS was the traveling salesman problem, a classic problem of wide practical application whose strong ties to production scheduling are well established (Baker 1974; Conway, Maxwell, and Miller 1970; and Lawler, et al. 1989). The large body of literature documenting effective classical approaches (as exemplified by Padberg and Rinaldi (1989)) provided testbed problems. Knox and Glover (1989) demonstrated the power of a prototype TS method on a benchmark set of seven "small but hard" symmetric traveling salesman problems ranging from 25 to 110 cities (Stewart 1987). In their approach, initial tours were randomly generated and diversification was achieved solely by repeated random restarts of the procedure. Given a complete tour, the commonly used 2-opt moves were employed as a basis for generating successive tours (where two non-adjacent edges are deleted and replaced by the unique pair of edges that reconstruct a tour).

The relatively small size of the problems (where the size is given by n, the number of cities) and the easy evaluation of the move value, i.e., the change in the tour length, for any of the $n(n - 3)/2$ possible moves apparently made it unnecessary to implement a candidate list where only a subset of admissible moves would be considered. The short term memory structure employed in their approach consisted of a tabu list of defined length where the oldest element of the list was replaced by the longer of the deleted edges of the most recent move. Thus, by prohibiting any member of the tabu list from becoming a tour element for a specified number of future moves, progress from a local optimum was allowed and reversals of recently executed moves were prevented. Earlier studies of different problem classes found that the length of the tabu list was insensitive to the problem being attacked and that a typical "good" length was in the neighborhood of 7 moves. However, in the traveling salesman problem context, it appeared that the best tabu list length was related to the number of cities in an approximately linear fashion, where the best list sizes were found to lie between $3n/2$ and $3n$. In a more recent study by Tsubakitani and Evans (1991), tabu list sizes in the range of $n/8$ to $n/2$ were found to generate the best results for problems with 20, 50 and 100 cities using a 2-opt based TS method. For a TS procedure based on 3-opt neighborhoods, they found the best tabu list size to range from $n/16$ to $n/8$.

In order to allow promising moves to override a tabu status, the simplest aspiration criterion is to allow a tabu move to be selected if it will lead to a strictly better solution than any found in previous moves. Knox and Glover implemented a somewhat more complex aspiration criterion where a tabu edge had its own aspiration criterion equal to the length of the tour from which the edge was dropped. A tabu edge was allowed to enter the tour if its entrance caused a tour length superior (smaller) to its aspiration criterion. Either of these aspiration procedures disallow immediate move reversals.

On the three problems with 25, 42, and 57 cities, the known optima were easily found with the 57 city problem requiring about five minutes on an IBM AT microcomputer. Although their prototype used only the short term memory component of tabu search, the results for the four larger problems yielded solutions superior to any found prior to that time. The method was also shown to be robust in regard to the parameter settings for the search. For example, runs on the 110 city problem, the largest and most difficult of the set, yielded improved solutions over previous methods for every setting attempted with run times ranging from 4 to 20 minutes on a VAX 11/780. Their study also included comparisons with specialized simulated annealing (Lam 1988) and genetic algorithms (Whitley 1989) approaches in existence at that time.

Malek, et al. (1989a, 1989b), building upon the work of Knox and Glover, developed both serial and parallel TS approaches to the traveling salesman problem. Their serial approach differed in two ways: (1) a simpler aspiration criterion was used where a tabu status was overridden only when the tabu move would yield a better solution than had been achieved to that point in the search and (2) a frequency based long term memory diversification strategy was incorporated. The long term memory recorded the number of iterations that each edge was a member of the tour. This information was used to penalize any restart random tour building process that attempted to include edges that were often present in the tour. Reported computational results included nine different

tabu condition attributes and nine different problems ranging from 25 to 100 cities. It was determined that the best length of the short term memory ranged from 7 to 30 for the best tabu condition which limited the movement of only one of the edges affected by the 2-opt move. The results of the studies showed that, in every instance, the TS method incorporating the long term memory in the restart procedure outperformed both the TS method with a simple random restart and a simulated annealing approach. For the larger problems, the new TS approach dramatically improved the best previous results.

In the parallel TS implementation of Malek, et al. (1989a), the long term memory diversification structure was removed in favor of an intensification process. At each restarting point, the best solution achieved since the last restart by any one of four parallel processors was used as the new restart solution. In each cycle, the four processors simultaneously executed the serial TS approach with different short term memory lengths and different tabu conditions. This parallel approach outperformed the serial method with respect to both time and solutions obtained. However, the total amount of processor-seconds, as opposed to absolute length of run time on the parallel machine, still favored the serial approach in this implementation.

Malek, et al. (1989b) use the serial TS method (without long term memory) and the simulated annealing method described in Malek, et al. (1989a) to form a hybrid simulated annealing-TS method. Using a parallel machine, they employed either two, four, or eight processors to run the serial TS and simulated annealing methods (in equal numbers) simultaneously. At each restarting point, the best solution achieved by any one of the parallel processors is used as the new restart solution. Solutions obtained by the hybrid method were, on the average, better than either the parallel TS or simulated annealing method. These were obtained at a total computational effort greater than that required by the TS approach. (It should be mentioned that in this case the term "hybrid" does not mean that elements from simulated annealing and tabu search are combined to create a new method, but rather that these two methods are run in parallel and share solutions.) Since the tabu search approach independently dominated the simulated annealing approach, this raised the question of whether the improvement resulted simply by virtue of giving the tabu search approach a larger variety of good restarts and whether the use of a probabilistic tabu search approach might still do better. Although noted, these questions were not examined.

Fiechter (1990) developed a parallel TS approach and attacked randomly generated problems with cities uniformly distributed on the unit square. The size of these problems ranged from 500 to 100,000 cities. The first tour is randomly generated and immediately partitioned into k open subtours or "slices," where each slice has one vertex in common with each of its two adjacent slices. The paths between all such pairs of slice defining vertices are "minimized" independently on separate processors by use of the 2-opt procedure. The slices are then uniformly shifted so that the vertices from the first set of slices are interior to the new slices and the slice minimizations are repeated. The best slices are then recombined. By breaking the problem into k subproblems, the set of possible 2-opt moves is greatly reduced, leading in effect to the implicit use of a simple candidate list.

Following the partitioning procedure, which provides a form of intensification, diversification is achieved in an adaptive manner through the use

of a sequence of special 4-opt moves called "super-moves" which cause the deletion of four tour edges and their subsequent replacement by complementary edges to rebuild the tour. This is done in such a way that an entire portion of the tour is displaced to another location. Such a tour modification would require at least four 2-opt moves, possibly selected in defiance of unfavorable changes to the tour. Each 4-opt move is selected as the best available within the context of a candidate list and a minimal required change in the tour. A TS framework is imposed upon the high level super-move search by keeping track of the eight vertices involved with each such move and not allowing subsequent moves that use more than a specified number of "tabu vertices."

At the end of the diversification sequence, the intensification procedure is once more implemented. This alternation between intensification and diversification is performed until a specified iteration limit is reached. Fiechter (1990) presents computational results both on two classical problems with about 500 cities and on a set of large randomly generated problems. The algorithm reportedly compares advantageously to a simulated annealing approach due to Troyon (1988) on the two classical problems and, measured by reference to a theoretical estimate provided by Beardwood, Halton, and Hammersley (1959), performs very favorably on the larger problems.

The foregoing traveling salesman problem efforts embodied the use of relatively primitive moves for transforming one tour into another. They are included in this overview for two reasons. First, they embody some of the earliest efforts to apply tabu search to problems that share features in common with scheduling problems. Such research provided learning experiences that laid a foundation for subsequent more advanced tabu search implementations.[Ψ] In the following sections we will indicate early implementations of flow shop and machine scheduling methods that similarly became the basis for more recent advances in these areas.

These experiments also disclosed the ability of tabu search to produce solutions of significantly higher quality than those obtained by applying the same types of moves without its guiding influence -- e.g., by using these moves in a classical descent method coupled with random restarting. As a result, this work motivated researchers to look at applying TS to problems whose structures embody more realistic features characteristically found in practical applications.

Work on the traveling salesman problem is ongoing. Advances continue to be made by tailoring special algorithms and moves to take advantage of the graph theoretic and polyhedral (cutting plane) properties of this problem. In the heuristic area, the study of Johnson (1990) has shown that a refined implementation of the compound moves proposed by Lin and Kernighan (1973) can produce exceedingly good outcomes, and these constructions have been made the basis of other promising approaches (Ulder et al (1991), Muhlenbein (1992)). An alternative line of research has resulted in new types of advanced moves for

[Ψ] The approach of Fiechter (1990) represents a transition from simpler to more advanced approaches, and includes several elements now often incorporated in modern TS methods. As noted, this work was remarkably successful in large scale applications involving as many as 100,000 nodes, in spite of relying predominantly on primitive 2-opt (and occasional 4-opt) moves.

traveling salesman problems, including a class of *generalized insert moves* by Gendreau, Hertz and Laporte (1990) that have been successfully extended and applied to vehicle routing (as we report in a subsequent section). Similarly, Glover (1991,1992) has proposed new search neighborhoods that derive from the notion of *ejection chains*. Such neighborhoods are designed to produce moves of greater power with an efficient investment of computer effort. Such a move generation process provides a *combinatorial leverage* effect which has the property of allowing the size of the neighborhood to grow exponentially while the effort of finding a best move in the neighborhood grows only polynomially. The approach of using neighborhood structures based on ejection chains within tabu search has been shown to be powerful in solving difficult generalized assignment problems (Laguna, Kelly, Gonzalez-Velarde, and Glover 1991).

Tabu search provides a framework that can govern the application of these more advanced types of moves as well as simpler ones, and the study of approaches that exploit such connections offers interesting possibilities for future research.

2.2 *Single Machine Problems with Linear Delay Penalties and Sequence Dependent Set Up Costs*

Laguna, Barnes, and Glover (1991, 1992) developed powerful TS methods for two related single machine problems with linear delay penalties and sequence dependent set up costs (Barnes and Vanston 1981) which, prior to their TS methods, had defied solution except for very small problems. The first problem studied is equivalent to an asymmetric traveling salesman problem for which the salesman is penalized for arriving late to each city. In the context of machine scheduling, the problem is described as follows. At time zero, n jobs simultaneously arrive at a continuously available machine. Each job i (i = 1, 2, ..., n) requires t_i units of time on the machine and a penalty p_i is charged for each unit of time that job commencement is delayed after time zero; s_{ij} is the setup cost of scheduling job j immediately after job i. Two dummy jobs, 0 and $n+1$, are included in every schedule, where $t_0 = t_{n+1} = 0$ and $p_0 = p_{n+1} = 0$. The $s_{0,j}$ and $s_{i,n+1}$ are considered initial and final cleanup costs. The objective is to find the schedule that minimizes the sum of the delay and setup costs for all jobs. If the setup costs are ignored, the problem is optimally solved using Smith's rule (i.e., sequencing the jobs by increasing time-to-penalty ratios). This rule will also provide a good heuristic solution when the contribution of the delay penalties is much larger than the contribution of the setup costs. If delay penalties are ignored, the problem becomes an asymmetric traveling salesman problem.

This problem, as the asymmetric TSP, is a "permutation problem" where we desire to find the best of the $n!$ possible sequences. The only available optimal scheduling algorithm for this problem is due to Barnes and Vanston (1981) who used the dynamic programming/branch and bound method of Morin and Marsten (1976). However, the explosive dimensionality of the problem caused their algorithm to fail for problems with more than 20 jobs.

Laguna, Barnes, and Glover's (1991) TS method starts from a solution generated by a constructive, deterministic heuristic from Barnes and Vanston (1981). Their best TS method simultaneously used two kinds of moves: (1) an

insert move which moved one job from its current position and inserted it between two other jobs and (2) a *swap* move which exchanged the positions of two specific jobs. In a defined set of candidate moves, the move with the most favorable objective function change was chosen at each step.

The most effective tabu restrictions developed were: (1) in *swap* moves, after a move of jobs i and j, where i is earlier than j, job i cannot be placed at or any earlier than its original position on subsequent schedules until the short term memory has expired and (2) in *insert* moves, the single moving job is not allowed to change position again until the short term memory has expired. The method uses a simple aspiration criterion in which a tabu move is allowed if its execution leads to a solution that is better than any other one visited before.

The efficient search for the "best" move at each search state is critical to most TS procedures. In this case, swap and insert moves define two $O(n^2)$ neighborhoods. Therefore, to limit the total number of moves considered for evaluation, a candidate list was created to include those exchanges that involved jobs no more than d positions apart, where d was set at $\lfloor n/2 \rfloor - 1$ (where $\lfloor x \rfloor$ is the largest integer less than or equal to x). The swap and insert move values were calculated using specially developed shortcut formulas and by introducing a memory scheme that isolated and updated only those move evaluations that changed from iteration to iteration.

The use of two classes of moves allowed two different "neighborhoods" to be searched, i.e., sequences easily reachable by swap (insert) moves were difficult to reach using insert (swap) moves. This markedly increased the performance of the search procedure. Experiments showed that with a single starting point and a constant short term memory size of 7, the TS approach easily achieved the known optimal solutions to problems with up to 20 jobs in an average time that did not exceed 15 seconds (on a 386-20 IBM PS/2 microcomputer). The restrictive nature of the tabu structure, embedded in this procedure, allowed a spectrum of short term memory sizes to be effective in solving larger problems (i.e., with up to 60 jobs).

Diversification of the search is often required to find good solutions to large problems where the procedures are expected to run for long periods of time. Laguna and Glover (1992) used an approach known as *target analysis* to add a diversification component to the foregoing TS procedure. This approach links the fields of artificial intelligence and operations research by giving heuristic and optimization procedures the ability to learn what rules are best for solving particular classes of problems. The target analysis methodology disclosed the regional dependency of good choice rules. Specifically, the goodness of a move was measured in two different ways depending on whether the search state was improving or non-improving. A frequency-based long-term memory function was employed to discourage frequently executed moves from being selected during non-improving states. Computational experiments empirically showed the robustness of the resulting TS method which was able to obtain the best known solutions to problems with up to 100 jobs using a single parameter setting. The research established the usefulness of target analysis as means for integrating effective diversification strategies, based on long term memory functions, within tabu search.

In Laguna, Barnes, and Glover (1992) a more difficult sequencing problem is addressed that includes sequence dependent setup *times*. The setup times may

be interpreted, in the traveling salesman context, as traveling time between cities. The salesman spends certain amount of time in each city (i.e., the processing time) and the traveling time from city to city depends on the sequence. The TS procedure to deal with this problem was entirely based on the one developed for the instances without setup times. The main differences are in the construction of starting solutions, the move evaluations, and the updating of the move values. The resulting procedure was able to find all known optima for problems with up to 22 jobs in less than 4 seconds on a 386-20 IBM PS/2 computer. For a problem with 24 jobs, the TS method considerably improved the best known solution (which had been found with a curtailed dynamic programming/branch and bound approach). The experiments were performed starting the procedure from a single point and using only one parameter setting. No more than 500 iterations were needed to match all known optima or improve the best known solutions.

2.3 A Single Machine Problem with Deadlines, Holding Costs, and Sequence Dependencies

Woodruff and Spearman (1992) formulate a general sequencing problem that includes two classes of jobs with setup times, setup costs, holding costs and deadlines. They use a TS method with insertion moves to transform one trial solution into another. Due to the presence of deadline constraints, not every sequence is feasible. However, the search path is allowed to visit infeasible solutions in an strategically oscillating fashion. A candidate list is also used as a means of reducing the computation effort involved in evaluating a given neighborhood. Diversification is achieved by introducing a parameter d into the cost function. Low values of d result in the selection of the best (with reference to the objective function value) available move as customarily done in a deterministic tabu search while high values result in a randomized move selection which resembles a variant of probabilistic tabu search. The tabu list designed for this approach is based on the concept of hashing functions. The list is composed of two entries for each visited sequence, the cost and the value of a simple hashing function (i.e., a value that represents the ordering of jobs in the sequence). Computational experiments were conducted on simulated data that captured the characteristics of the demand and production environment in a large circuit board plant. For a set of twenty test problems, the average objective function quality was 0.97 and the optimal objective function was achieved in seventeen cases. These results showed the ability of the TS method in finding near optimal solutions to this difficult sequencing problem. This study also marks the first application of TS where hashing functions are used to control the tabu structures. A more detailed study on this kinds of functions and their use within the TS framework is given in Woodruff and Zemel (1992).

3. Multiple Machine Scheduling Problems

In the area of production scheduling on multiple machines, tabu search has also enjoyed a remarkable amount of success. This section presents the TS based research that has been applied to seven types of multiple machine production scheduling problems.

3.1 The Weighted Flowtime Problem

Barnes and Laguna (1992) extended their single processor approach to attack a classical production scheduling problem on parallel machines, the *weighted flowtime problem* (Barnes and Brennan, 1977). In the weighted flowtime problem, N single-operation jobs simultaneously arrive at time zero to be processed by M continuously available identical machines. Job i requires t_i units of time on any machine. A penalty, p_i, is charged for every unit of time that the commencement of job i is delayed. The time until job i is started is the sum of the processing times of all jobs scheduled earlier than job i on the same machine. Jobs are indexed by Smith's rule (in *natural order*), no technological constraints restrict job sequences, and no preemption is allowed. A schedule is any partition such that every job is assigned to a machine. Since it is well known that at least one optimal schedule will have the jobs on each machine ordered in natural order, the problem may be viewed as finding the optimal partition of *n* jobs into *m* sets (where *m* is the total number of machines). The objective of this optimization problem is to minimize the total delay penalty for all jobs.

This TS method starts from a solution generated by a one-pass construction heuristic (referred to as H_1) due to Baker (1974). The procedure then performs insert and swap moves to perturb the current trial solution. Swap moves allow a pair of jobs resident on different machines to trade positions on their respective machines. Insert moves allow a single job to be transferred from one machine to another. The tabu restriction on swap moves prohibits movement of one of the jobs during the extent of the short term memory. A swap move is a candidate move if it preserves the natural order sequence on both machines. The constant-cardinality limitation of the swap move is overcome by the simultaneous consideration of insert moves. An insert move is a candidate move if it preserves natural order on the receiving machine. The controlling factor on the size of the composite candidate list of moves is the preservation of natural order in each machine.

Diversification strategies are particularly helpful when better solutions can be reached only by crossing barriers or "humps" in the solution space topology. A simple interpretation of this concept led to incorporating an effective diversifying element in the TS procedure. Since swap moves do not modify the cardinality of the partition in the current solution, they are considered to have a smaller move distance than insert moves. Therefore, in addition to the traditional decision rule for selecting the best move at each iteration (i.e., the admissible move with the smallest move value), the selection of non-improving swap moves was forbidden after an initial period where the search "settles down." This strategy has the effect of intelligently perturbing the current solution, while escaping from a local optimum, to an extent that the search is directed to a region that is different than the one being currently explored. The implementation of this strategy as applied to experimental problems resulted in about an order of magnitude improvement in the efficiency of the method. A number of experiments were conducted to compare the performance of the TS method with the best available exact procedure (a branch and bound developed by Barnes and Brennan 1977). The comparisons were based on the time that each

procedure took to first find the optimal solution. The TS method was able to find the optimum in at least an order of magnitude less time than the branch and bound algorithm, which was designed to quickly find good solutions to assist in fathoming the nodes of the implicit enumeration tree.

3.2 The Multiple-Machine Weighted Earliness Problem with Deadlines

Laguna and Gonzalez-Velarde (1991) applied TS to the *multiple-machine weighted earliness problem with deadlines*, using GRASP (Feo and Resende 1989) to provide diverse initial solutions, and branch and bound to optimize the sequence of jobs assigned to each machine. In this problem, n jobs are to be processed by m parallel machines in order to meet deadlines and minimize the total earliness penalty. After an initial solution is generated by the first phase of the GRASP approach, which consists in randomizing the selection step of a greedy heuristic, a tabu search was performed using a combination of swap and insert moves. The search path was only allowed to visit feasible solutions, therefore the candidate list of moves was restricted to those that preserved feasibility (i.e., moves that did not violate the deadline constraints). In addition, the candidate list was constructed to exclude moves that were expected to have unreasonably large deteriorating values. The concept of *essential moves* was introduced and used within a choice rule that was only activated in non-improving regions. In general terms, an essential move is such that its execution is considered necessary if an improved solution is to be found later in the search. The use of this concept provided the search with a very powerful diversification mechanism.

After a tabu search had been performed from a prespecified number of starting points, the best solution found was subjected to a branch and bound postprocessing. Each machine was individually optimized according to the assignments given by the best solution found during the tabu search. Computational experiments showed that the merit of the postprocessor increases with the number of jobs. These experiments also showed that the final solution obtained with this hybrid approach is about 10% better than the best deterministic one-pass heuristic known.

3.3 Vehicle Routing Problems

Like the traveling salesman problem, vehicle routing problems (VRP) are also closely related to production scheduling (Lawler, Lenstra, Rinnooy Kan, and Shmoys, 1990). Semet and Taillard (1991) present a real-life VRP that includes not only capacity and time-window constraints but also a number of restrictions imposed by different vehicle attributes and access to customers. The problem deals with orders (in the range of 70 to 90) placed by 45 stores located in the cantons of Vaud and Valis in Switzerland. In the context of production scheduling, the problem may be viewed as one where the machines are non-identical in terms of both available capacity and capability. A heuristic solution to the problem, based on a standard VRP method and a clustering algorithm, is given as an initial approach (this technique is currently used by the Swiss chain interested in solving the VRP). The heuristic works at two levels: first the large vehicles (truck-trailer combination) are assigned to accessible customers, and then the small vehicle (i.e., the truck) is used to serve inaccessible customers

(i.e, those that are not able to accommodate a trailer) in the vicinity. This may be interpreted as assigning a large job (or project) to a particular facility and requiring that this facility performs the related tasks (e.g., subassembly operations) that are necessary to complete the project. The objective function is given as minimizing the total distribution cost. The TS method implemented by Semet and Taillard (1991) uses simple insertion moves (i.e., changing one order from one route to another) as the main mechanism to explore the solution space. Although the mechanism is simple in nature, move evaluations are extremely expensive. In fact, it was determined that 90% of the running time was spent on finding optimal assignments of vehicles to routes. The method was accelerated by using relaxed evaluations of trial solutions, by reducing the size of the neighborhoods, and by aggregating orders (so the size of the problem is in fact reduced). It is interesting to note that for this problem a more restrictive tabu structure seemed to give the better results. This structure places tabu restrictions on customers as opposed to the individual orders generated by these customers. The TS method uses a tabu list size that is randomly generated every 60 iterations from a uniform distribution with parameters 6 and 30. The authors predict 10 to 15% savings in total distribution costs from solutions generated with their TS approach.

Gendreau, Hertz, and Laporte (1991) present another application of tabu search to the VRP where capacity and route length restrictions are considered. This is analogous to a production scheduling problem where the machines have capacity restrictions and the makespan may not exceed a given limit. The described TS method employs insertion moves to direct the search from one solution to another where the moves are performed using the GENI (generalized insertion) procedure (Gendreau, Hertz, and Laporte, 1990). After a move is performed, it is classified tabu for a *random* number of iterations. This implicit implementation of randomly changing tabu list sizes requires a minimum amount of bookkeeping. Computational experiments with 14 benchmark problems with number of customers ranging from 50 to 199 show that the TS procedure developed in this research effort outperforms all existing heuristics. The authors attribute their success to strategically oscillating between feasible and infeasible solutions and to the diversification power embedded in the GENI procedure.

More recently, Osman (1992) has developed a TS procedure for the VRP that uses a different move mechanism but that also obtains results of comparable quality.

3.4 The Permutation Flow Shop Problem

One of the early applications of TS in scheduling is due to Widmer and Hertz (1989), who compared TS to six alternative heuristic approaches to the permutation flow shop problem. The permutation flow shop problem (French 1982, Lawler, et al. 1989) has n multiple operation jobs arriving at time zero to be processed in the *same order* on m continuously available machines. The processing time of job i on machine j is t_{ij} and individual operations are not preemptable. The objective is to find the ordering of jobs that minimizes the makespan, i.e., the completion time of the last job.

Widmer and Hertz (1989) used a simple insertion heuristic based on a

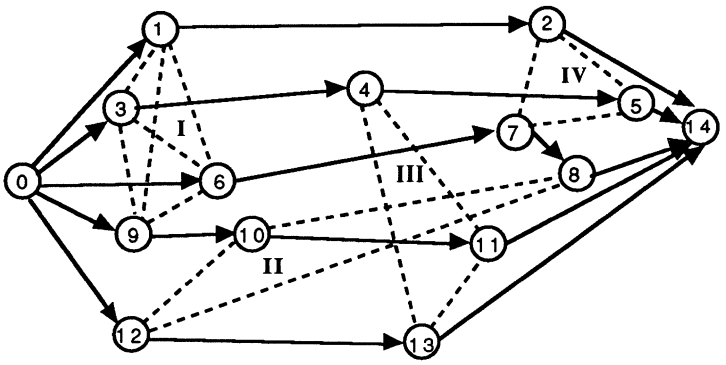
traveling salesman analogy to the permutation flow shop problem to generate the first ordering of the jobs. Next they implemented their TS approach which considers a single type of move where jobs i and k in positions x and y exchange their positions in the ordering. At each iteration, the best non-tabu move was executed. The tabu list was exclusively set at a length of 7 moves and job i (k) was not allowed to return to position x (y) for the duration of its tabu status. The process continued for a specified number of iterations and then terminated.

The results of a set of computational experiments with their TS approach is compared with six previously developed heuristic methods. In experiments on 50 problems with n and m ranging from values of 5 to 20 where the maximum number of TS iterations was set to $n+m$, the results are mixed. In direct competition with the best previous heuristic developed by Nawaz, Enscore and Ham (1983), the TS method returned superior solutions in 58% of the problems and obtained the best solution found in 92% of the problems. These successes were achieved at the expense of greater computational effort. Note that the TS procedure included neither an intensification nor diversification phase (i.e., other than the ones implicitly provided by the short term memory component) and no discussion was given on the efficiency of the move evaluation method or the use of an aspiration criterion. Nevertheless, the approach is noteworthy for being the first TS study for this problem class.

3.5 The Jobshop Scheduling Problem

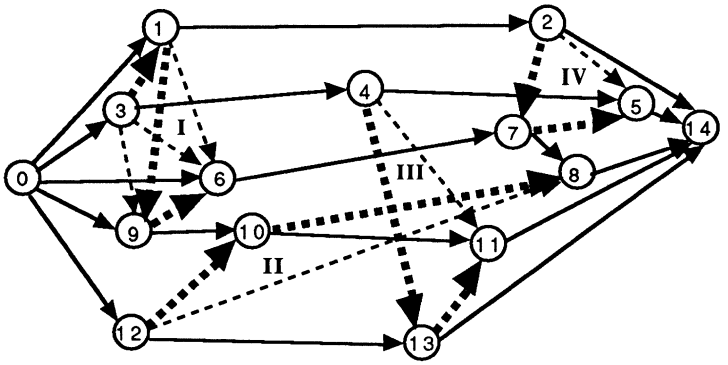
Taillard (1989) constructed a TS approach to the jobshop scheduling problem (French 1982, Lawler, et al. 1989) using the classical disjunctive graph formulation (Roy and Sussman 1964; Balas 1967, 1969). Figure 1 presents a small problem with 5 jobs and 4 machines. As may be observed, the technological precedence ordering of the job operations is enforced by directed paths of conjunctive arcs (solid lines) in the graph. The order of the operations that are performed by each machine are detailed by an acyclic *selection* of the direction of the disjunctive edges (dashed lines) for each machine clique. Figure 1(a) presents the graph prior to selection and Figure 1(b) after a feasible selection has been made. To minimize the makespan, we must find the selection which yields the directed graph with the minimal longest path. Taillard took advantage of two basic structural properties of such graphs: (1) reversal of a single directed disjunctive arc on the critical maximum length path will not form a cycle and (2) to improve a schedule at least one such disjunctive arc must be reversed. Thus, a move was defined to be a reversal of a critical disjunctive arc and the value used for move selection was the *projected* change in the makespan, Δ_h, as given by Balas (1969). Δ_h gives the actual change in the makespan when $\Delta_h \geq 0$. However, $\Delta_h < 0$ gives only a lower bound on the makespan change. The projected move values were computed using the o(v) longest path algorithm described in Adams, Balas, and Zawack (1988), where v is the total number of operations.

Taillard randomly selected the length of the short term memory from a stated range of values and the length was changed each time the number of subsequent moves marginally exceeded that short term memory length. This avoided the problem of identifying the best short term memory and helped to



Job	Operations	Machine	Operations
1	0,1,2,14	I	3,1,9,6
2	0,3,4,5,14	II	12,10,8
3	0,6,7,8,14	III	4,13,11
4	0,9,10,11,14	IV	2,7,5
5	0,12,13,14		

(a)



(b)

Fig. 1 An Example of a Disjunctive Graph Representation of a Jobshop Scheduling Problem

avoid cycling and the condition where no moves were available. The values for the tabu list length were made proportionally larger as the total number of operations increased. For a 10 machine, 10 job problem the tabu length ranged from 8 to 14. Taillard implemented the simplest aspiration criterion of allowing a tabu move only if it leads to the best makespan found to that point in the search. No attempt was made to incorporate intensification, or diversification procedures, or the use of more than one kind of move. Taillard gives no description on how starting solutions were obtained.

Taillard compared his results against the shifting bottleneck method of Adams, Balas, and Zawack (1988), and against the simulated annealing method of van Laarhoven, et. al. (1988). However, the only comparisons presented in the paper are in terms of the mean optimal solution for a selected set of 10 machine, 10 job problems Taillard found that the TS method was uniformly superior to simulated annealing and was also better than the shifting bottleneck method in terms of solution quality and computational effort on a set of 10 by 10 problems. However, mention is made of cases on larger problems where the shifting bottleneck method achieved better solutions. (This suggests that superimposing TS onto the shifting bottleneck method might yield superior results on larger problems.) Taillard states that the TS methods were much easier to implement and the inherent flexibility of TS approaches would allow other types of problems, such as the jobshop problem with dynamic set up times, to be attacked with only slight changes of implementation.

Barnes and Chambers (1992) also developed a TS approach to the jobshop scheduling problem. They obtained an initial solution by using the *active* and *non-delay* forms of seven dispatching rules (Baker 1974; Conway, Maxwell, and Miller 1970), i.e., SPT, LPT, LWKR, MWKR, MWKR-SPT, MOPNR, MWKR/P. Having computed all 14 solutions, they selected the resulting schedule with the minimal makespan as their starting solution. Like Taillard, they used the disjunctive network formulation and a move was performed by the reversal of a single directed disjunctive arc on a critical maximum length path. All critical paths and their associated critical arcs were directly considered. At each iteration, they selected the move that gave the best value of Δ_h (Balas 1969). The o(v) longest path algorithm of Adams, Balas, and Zawack (1988) made this evaluation process computationally feasible. (They stated that it was necessary to allow zero valued moves for the search to be successful. Not allowing zero valued moves caused early apparent success that resulted in inferior final schedules.)

After the initial solution was obtained, a contiguous range of short term memory values was selected. The initial search was conducted using the largest short term memory in the selected range. During the initial search, each time the best solution was improved, the associated schedule was stored on a push-down list for later retrieval. The initial search was terminated when a specified number of moves had been made with no improvement in the best solution.

At that point, a list of monotonically improving solutions was resident where, within the limitations of allowed search time, each would be used as an alternative starting solution for subsequent searches. Upon completion of the initial search, all tabu restrictions were removed and the best solution on the push-down list was popped off of the top of the list and installed as the starting solution for the next set of searches. Starting with the smallest short term

memory in the specified range, the search procedure was performed until a specified number of moves had been made with no improvement in the best solution found relative to the current search. This was repeated for each short term memory value in the specified range.

Each time the overall best solution was improved, the associated schedule was stored on the push-down list. Once all short term memory values had been applied to a particular starting solution, that solution was never used again. In this manner, diversification was achieved through restarting the search at the various solutions where the search was then intensified by conducting the search with the prescribed set of short term memory values.

The procedure described above was implemented until the prescribed maximum search time was reached. Investigations for a spectrum of contiguous short term memory ranges allowed identification of one or two best ranges for each problem size that was investigated.

It was not rare for the search to reach a situation where no non-tabu moves were available. This could have caused a premature termination of the search. Barnes and Chambers combatted this with a simple strategy. When no moves were available, the search was restarted with the current solution with all tabu restrictions removed.

Barnes and Chambers presented computational results for 31 test problems where other heuristics had been applied (Lawrence 1984; Adams, Balas, and Zawack 1988; Applegate and Cook 1991, Applegate, 1991). These problems ranged in size from 10 machines and 10 jobs to 15 machines and 20 jobs and included 7 larger "open" problems and 24 problems where the optimum is known. (As was found by Taillard (1989), problems where the number of jobs was significantly greater than the number of machines or when the number of machines is small present no challenge to the TS approach.) Barnes and Chambers' technique consistently outperformed the bottleneck method of Adams, Balas, and Zawack (1988) on all the problem sets and found the optimum in 13 of the 24 cases where the optimum was known. Indeed, the superior performance was particularly evident where, in every case, Barnes and Chambers' method improved the best known solutions from Applegate and Cook (1990) on the 7 open problems.

Dell'Amico and Trubian (1992) developed a more complex tabu search based approach to the jobshop scheduling problem which also uses the disjunctive network formulation described above. Dell'Amico and Trubian generate a single starting solution for each run of their method using a bidirectional dispatching technique similar in several ways to the unidirectional methods adopted by Barnes and Chambers. Different starting solutions for multiple runs are achieved by an embedded randomization procedure within the bidirectional dispatching technique.

They investigated five different types of moves (or neighborhoods). The first neighborhood, N1, was used by Taillard (1989) and by Barnes and Chambers (1992) and the second neighborhood, N2, was that suggested by Matsuo, Suh and Sullivan (1988). The third type of neighborhood, NA, was based on N1 where adjacent operations i and j are on the same machine and arc (i,j) is on the critical path of the disjunctive network. Let PM[i] be the machine predecessor of i and SM[j] be the machine successor of j. A move considers, for all critical path arcs, all possible permutations of {PM[i],i,j} and {i,j,SM[j]} where arc (i,j) is inverted. Dell'Amico and Trubian (1992) show that the neighborhood thus

defined can be analysed in O(n) operations and always yields a feasible solution. Further, they prove a connectivity property that an optimal solution can be reached, in a finite number of moves, from any feasible solution using only NA.

The fourth neighborhood, NB, uses a critical sequence, *CS*, of adjacent operations on the same machine such that all operations are on the critical path of the associated disjunctive network. For all operations (nodes) $x \in CS$ (and for all such critical sequences), move x to the start or end of CS, disallowing any infeasible solutions that result. Dell'Amico and Trubian do not perform an exact feasibility test but rather used a simple rule that precludes infeasible moves and a particular type of feasible but nonimproving moves. They also show that NB possesses the connectivity property described above. The fifth and final neighborhood considered, NC, combines NA and NB and is the neighborhood definition that was empirically determined to be the superior move structure for their TS method.

The attributes by which prospective moves associated with NA and NB are deemed tabu are related to the component "single" swaps of adjacent operations that would lead to the new solution achieved by the move. Any move which contained an elemental tabu single swap is tabu. The short term memory length is dynamically defined depending on the current search status. If the move just performed has led to a makespan better than any before, the short term memory length is set to 1, i.e., the current move may be reversed after only one additional move is performed and previous tabu restrictions are ignored. If the last move improved the makespan and the short term memory length is greater than a prespecified minimum value, *MIN*, reduce the short term memory length by one. If the last move did not improve the makespan and the short term memory length is less than a prespecified maximum value, *MAX*, increase the short term memory length by one. MIN and MAX are used to control the search by combatting cycles in the search and by not allowing too many moves to be tabu which would result in an overly restrictive search. After a specified number of moves, the values of MIN and MAX are randomly reset to new values selected from prespecified ranges.

The simplest of aspiration criteria is employed, identical to that used by Barnes and Chambers (1992). When the procedure is blocked by the absence of non-tabu moves, they continue the search by randomly selecting a move from those available. Whenever the search has performed a preset number of moves, δ, without improvement, the search is restarted at the best previous solution. This type of restart-intensification procedure is allowable because of the nondeterministic nature of their method.

The termination criterion is tied to the performance of a block of δ moves. If, at the end of the block, a preset number of moves, MAXITER, has been reached and no global improvement has been achieved in the last δ moves, stop.

Dell'Amico and Trubian present computational results for a single setting of the large set of parameter values possible in their method as applied to 53 problem instances taken from Muth and Thompson (1963), Adams, Balas, and Zawack (1988), and Applegate and Cook (1990). The 31 test problems studied by Barnes and Chambers (1992) were included as a subset. The remaining 22 problems have already been shown to be unchallenging by Taillard (1989). Because of the nondeterministic structure of their method, they ran their method five times for each problem with different random starting solutions.

Dell'Amico and Trubian's method found the optimum in 11 of the 24 cases where the optimum was known and improved the best known solutions from Applegate and Cook (1990) on all 7 open problems. On the 20 problems where Dell'Amico and Trubian's method did not find the optimum, it found a better makespan than the Barnes and Chambers approach on 11 of the problems and an inferior solution on 9 of the problems.

3.6 A Job Shop Scheduling Problem with Tooling Constraints

In the classical job shop problem, one of the main assumptions is that the setup times for the operations are sequence-independent and are included in the processing times. When tooling constraints are taking into account, as required in scheduling models for flexible manufacturing systems, the above assumption must be removed. Widmer (1991) applies tabu search to the solution of the *job shop scheduling problem with tooling constraints.* His method consists of a basic implementation of the short-term memory component of tabu search with a single list of fixed size. Insertion moves are evaluated with reference to a penalized cost function that assigns weights to the makespan, the number of tool changes, the number of late jobs, and the number of oversteppings (i.e., the number of machines which are not able to complete their work during the planning horizon). The TS method is stopped after a number of iterations have been done without improving the incumbent solution. For his computational experiments, Widmer (1991) generated 100 problems with the number of machines ranging from 1 to 10, with up to 10 products, and 2 different lot sizes. Each problem was run 10 times from different staring points and the best solution was recorded. The results showed that the frequency with which the best solution is found on a single run decreases as the problem size increases (in both number of machines and products). Although Widmer's results can only be considered satisfactory in terms of speed and average performance, his work offers a challenge for incorporating more sophisticated mechanisms and strategies into his basic initial approach.

3.7 The Flexible Job Shop Scheduling Problem

Brandimarte (1992) developed two prototype hierarchical TS approaches to the *flexible* job shop scheduling problem, an extension of the *classical* job shop scheduling problem which leads to a much more difficult problem. Instead of one uniquely identified machine, a set of nonidentical machines can perform each of the operations on each of the jobs. Brandimarte chose to model the problem as two subproblems:

(1) a routing subproblem where each operation is assigned a unique machine from the set of capable machines
(2) a scheduling subproblem where operations assigned to each machine are sequenced.

A natural hierarchy is present between the two subproblems; once the routing problem is solved, the scheduling subproblem is a classical job shop scheduling problem. Brandimarte first developed a *one-way* approach where the

routing problem was solved by dispatching rules and the resulting job shop scheduling problem was solved using a tabu search method like that of Taillard with three randomly defined and limited move neighborhoods.

He next developed a superior two-way approach which iterates between the two subproblems. Brandimarte used a TS method very similar to that of Taillard (1989) to attack the job shop scheduling problem. His TS method for the routing problem made use of critical path information gathered from the solution of each job shop scheduling problem generated. A limited set of computational results were presented. However, as Brandimarte states "The aim of the paper is not to assess the efficiency of the proposed architecture, ... but to stress the advantages of tabu search from the point of view of software development effort and real world applicability." Brandimarte concludes his paper with a several suggestions for refinement and future research.

4. A Synthesis of the TS Methods in Production Scheduling

The TS procedures presented above contain both a number of similarities and a number of differences. The similarities come from the strategies that are used to create the basic tabu search procedures, while the differences are the actual mechanisms that are designed to implement these strategies. Table 1 summarizes the different characteristics of the TS methods described in this paper. The "Moves" column gives the procedure that was used for local search. The most popular move mechanisms are inserts and swaps (or 2-opt in the traveling salesman problem context). Insertion procedures are often preferred because a single swap may be achieved by two insert moves. Also, in the context of sequencing and partitioning problems, insert moves provide a higher degree of perturbation of the current solution than swap moves. Other move mechanisms have a larger dependency on the problem structure, e.g., the reversal of a critical disjunctive arc used by Taillard (1989) and by Barnes and Chambers (1992). More sophisticated neighborhood structures are achieved by procedures such as *ejection chains*, *GENI*, or by the *critical sequence approach* used by Dell'Amico and Trubian (1992). We should expect to see more of these kinds of neighborhood definitions, as researchers attempt the solution of even harder production scheduling problems via tabu search. The design of clever move mechanisms is a crucial step in the TS methodology, since it will have the greatest influence on the computational complexity and the searching power of the resulting procedure. The work by Evans (1987) examines the effect of neighborhood structures on search efficiency, and under some assumptions, bounds are calculated for the probability that a local optimum is global. Evans compares the search efficiency of move mechanisms such as *adjacent* and *all* swap exchanges in the context of the mean tardiness problem. His study suggests that using adjacent swaps as the main mechanism and examining the all-swap neighborhood every so often (or when at local optimum) would be desirable.

Different approaches have been taken to the issue of defining and implementing tabu lists. The "List Size" column in Table 1 shows some of these approaches. A simple short term memory function generally employs a single list of fixed size. The size of the list may or may not be a function of the problem size. For methods were the tabu structure is highly restrictive (e.g.,

Table 1. Summary of tabu search characteristics.

TS Method	Moves	List size	Attributes	Diversification	Intensification
Knox & Glover (1988)	2-opt	fixed f(problem size)	edges	random re-starts	none
Malek (1989a)*	2-opt	fixed	edges	frequency counts, random re-starts	re-starts from best
Fiechter (1990)*	2-opt	fixed	vertices	super-moves	tour partition
Glover (1991)	ejection chains	fixed & dynamic	nodes	frequency based	none
Widmer & Hertz (1988)	swaps	fixed	jobs	none	none
Laguna, Barnes, & Glover (1991 & 1992)	swaps, inserts	fixed	jobs, positions	none	none
Laguna & Glover (1992)	swaps, inserts	fixed	jobs, positions	penalize frequent non-improving moves	none
Woodruff & Spearman (1992)	inserts	hashing functions	sequences	random cost multiplier, strategic oscillation	none
Semet & Taillard (1991)	inserts	dynamic (random)	customers, orders	none	neighborhood reduction & aggregated demand
Gendreau, Hertz, & Laporte (1991)	GENI**	multiple random	customers	strategic oscillation	none
Barnes & Laguna (1992)	swaps, inserts	fixed	jobs	prohibits non-improving swaps	none
Laguna & Gonzalez-Velarde (1992)	swaps inserts	fixed	jobs	GRASP starts & essential moves	branch-and-bound
Taillard (1989)	arc reversal	dynamic (random)	arcs	none	none
Barnes & Chambers (1992)	swaps	dynamic contiguous range	positions	push down list	restarts at members of list
Dell'Amico & Trubian (1992)	permutations, critical sequence	dynamic, adaptive, random range	elemental tabu swaps	random re-starts	re-starts from best
Widmer (1991)	inserts	fixed	jobs	none	none

*Parallel implementation. **Generalized insertion procedure.

when jobs are being "locked" into their current positions), short list sizes are effective for a wide range of problem sizes. When the size of the tabu list changes during the search, the list is said to be *dynamic*. The changes of size may be achieved in a deterministic fashion, for example, by following a prespecified list of values. Alternatively, the size of the list may be changed randomly. In some applications, the size is chosen from a uniform distribution whose parameters are given by the user (Taillard 1989; Semet and Taillard 1991). Dell'Amico and Trubian (1992) use a dynamic and adaptive tabu list size which responds to the search status and is controlled by randomly selected upper and lower bounds. A different form of dynamic list is created by assigning a random tabu time to the most recently executed move. This approach in fact creates multiple lists of dynamic sizes (Gendreau, Hertz, and Laporte 1991). Finally, hashing functions may be used to keep track of previously visited solutions with a moderate investment of CPU time. A tabu list is constructed by storing the objective function values and the values from the hashing function. Moves that lead the search to solutions that match both values on the list are classified tabu. The size for this kind of tabu list tends to be large in proportion to the total number of iterations. The rationale is to prevent the search from visiting the same solution twice and avoid the possibility of cycling. However in tabu search, re-visiting solutions does not define a repeating cycle, unless the state of the search (i.e., the state of memory functions, tabu restrictions, etc.) is the identical every time the same solution is reached. Barnes and Chambers (1992) make use of this fact to implement an effective intensification strategy for the job shop scheduling problem.

The "Attributes" column in Table 1 indicates the kind of tabu restriction being enforced. In the TSP context, it is more restrictive to use *nodes* as the attributes than *edges*. In production scheduling it is common to use job indexes as the attributes. However, a number of alternatives for tabu restrictions are possible. For example, after a swap move is performed: (1) both jobs may be forbidden from moving, (2) the job that move to an earlier (later) position in the schedule may be locked into its new position, (3) both jobs are forbidden to exchange their positions, but they may participate in swaps with other jobs, or (4) the jobs (or just one job) may not be allowed to return to the positions they occupied prior to the exchange. The selection of the "right" tabu restriction is highly dependent on the other components of the TS method. As a general rule, however, restrictive tabu structures work well coupled with a diversification strategy.

The "Diversification" column shows the mechanisms used to encourage the search procedures to explore new areas of the solution space. Restarting the search from a randomly generated point is a common, though rather unsophisticated, way of diversifying. This may be done by generating a "completely" random solution, or by a "biased" random procedure (such as the first phase of GRASP). The type of diversification more generally prescribed by tabu search, but which has not yet been widely explored, is to use information generated during the search to bias the choices during a construction phase, and therefore increase the probabilities of generating a starting point that is significantly different than those solutions that have been already visited.

A different approach to diversification consists of altering the choice rule or the neighborhood structure during the search. The advantage of this approach is that the search is not stopped and no overhead (like an initialization step) is paid

for restarting it. The choice rule, for example, may be altered to penalize frequently executed moves. The frequency of the moves is recorded throughout the search on a long term memory structure. Alternatively, more than one neighborhood definition may be used during the search, basing the choice on long term memory information.

Although intensification strategies (shown under "Intensification" in Table 1) have not been extensively used, they are an important component of tabu search. The main challenge in implementing an intensification procedure is to identify when the search state is in a "good" region. Some evident forms of intensification are: (1) re-starting the search from the best solution found so far, (2) partitioning the best solution found so far and searching the neighborhoods of the resulting subproblems, (3) locking some jobs in the best (or current) solution into their best (or current) positions and performing the search on the free jobs, (4) using frequency information to construct a starting point with jobs occupying their "preferred" locations, or (5) keeping a list of "good" solutions and returning to them later in the search . Additional forms of intensification and diversification approaches, and ways that they may usefully interact, are discussed in Glover and Laguna (1992).

From the wealth of previous work that has been recounted above, a number of observations can be made about what constitutes the primary components of effective production scheduling TS method:

(1) Like most optimization methods, it appears that a procedure for obtaining one or more "good" starting solutions, as opposed to a random start, is beneficial to the eventual success of the TS method (particularly for larger problem instances). This does not appear to be as dominant a consideration for TS as it does for other methods. However, the use of more advanced TS intensification and diversification strategies for generating good "restart" solutions may result in advantages that have not yet been brought to light.

(2) The efficient selection and evaluation of moves is, perhaps, the single most important consideration in the successful application of TS to a production scheduling problem. The development of such an evaluation procedure should include an in-depth study of the problem's particular "structure," exploiting any possible streamlining of move evaluation. In conjunction with this effort, it is essential, especially in larger problem instances, to formulate a candidate list for move selection. This can range from simply limiting the search neighborhood about the current solution to identifying problem characteristics that exclude moves from the candidate list.

Target analysis methods (Laguna and Glover 1991) and other non-objective function oriented move selection criteria also fit into this realm of considerations.

(3) The most successful methods are likely to use more than one class of moves, either alternating their use or simultaneously considering them at each iteration. The use of two (or more) moves types enhances the

search process in at least two ways. First, a larger search neighborhood is explicitly considered, making favorable moves more likely. Second, the capability of embedded search diversification (Fiechter 1990; Glover 1991; Laguna, Barnes, and Glover 1991) either implicit or explicit, is made available.

(4) Attention to the best value of short term memory length, especially for larger more difficult types of problems, is essential. Unfortunately, the possibility of a universally applicable short term memory length has been discounted in the work cited above. However, statistical analysis of the kind reported in Tsubakitani and Evans (1991) provides useful information in determining the best list size for particular applications and neighborhood definitions. As illustrated by Dell'Amico and Trubian (1992), the best short term memory length may be a function of the current search status, forcing the developer of the TS approach to develop a controlled adaptive strategy to modify the short term memory length.

(5) Diversification strategies are essential to successful TS approaches to large production scheduling problems. This might take the form of a static long term memory array like that employed by Malek, et al. (1989a). However, it appears that a more effective technique is to employ intelligent dynamic diversification strategies like the "super-move" approach of Fiechter (1990), the region-dependent choice rule approach employed by Laguna and Gonzalez-Velarde (1991), or the diversification parameter used by Woodruff and Spearman (1992).

(6) Improvements have sometimes been realized by forming hybrid methods where TS is combined with other approaches such as simulated annealing, GRASP, and branch and bound (Laguna and Gonzalez-Velarde 1991; Malek, et al., 1989b).

5. Suggested Directions for Future Research

Few areas of interest are richer in problem types than production scheduling and only a few have been addressed, thus far, using TS methods. The successful TS implementations presented in this paper show the potential of this approach in the area of production scheduling and suggest the possibility for constructing hierarchical procedures for the solution of more complex problems. It may in fact be feasible to develop general TS methods capable of obtaining good solutions to entire families of production scheduling problems. The flexibility of the tabu search framework is useful in this connection, and conceivably may be exploited to design a "general purpose" computerized scheduler.

Another direction for future research is given by considering the superimposition of tabu search onto existing successful heuristics for specific scheduling problems. Many of these heuristics are myopic in nature, and therefore they generally become trapped in local optimal points. The TS methodology is well suited for allowing these heuristics to search beyond the first locally optimal point.

Finally, we note that parallel processing is an important and highly

inviting area for developing tabu search implementations. Strategies for intensification and diversification take on a special significance when coordinated in a parallel processing environment, and offer a variety of possibilities for exploration.

Bibliography

Adams, J., E. Balas and D. Zawack, "The Shifting Bottleneck Procedure for Job Shop Scheduling", *Management Science*, vol. 34, no. 3, March, 1988.

Applegate, D. and W. Cook, "A Computational Study of the Job Shop Scheduling Problem," *ORSA Journal on Computing*, Vol. 3, No. 2, 1991.

Applegate, D., Personal correspondence, 1991.

Baker, Kenneth R., *Introduction to Sequencing and Scheduling*, John Wiley Publishers, (1974).

Baker, K.R. and G.D. Scudder, "Sequencing With Earliness and Tardiness Penalties: A Review", *Operations Research*, vol. 38, pp. 22-36, 1990.

Balas, E., "Finding a Minimaximal Path in a Disjunctive PERT Network," *Theorie des Graphes, Journees internationales d'tude, Rome, Juillet 1966*, Dunod, Paris - Gordon and Beach, pp. 21-30, New York, 1967.

Balas, E., "Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm", *Operations Research*, vol. 17, pp. 941-957, 1969.

Barnes, J.W. and J. Brennan, "An Improved Algorithm for Scheduling Jobs on Identical Machines," *AIIE Transactions*, vol. 9, no. 1, March, 1977.

Barnes, J.W. and L. Vanston, "Scheduling Jobs with Linear Delay Penalties and Sequence Dependent Set-Up Costs," *Journal of Operations Research*, vol. 29, no. 1, January-February, 1981.

Barnes, J.W. and M. Laguna, "Solving the Multiple-Machine Weighted Flow Time Problem Using Tabu Search", *IIE Transactions*, in press, 1992.

Barnes, J.W. and J. Chambers, "Solving the Job Shop Scheduling Problem Using Tabu Search", Technical Report Series, Graduate Program in Operations Research, The University of Texas at Austin, ORP91-06,1992.

Beardwood, J., J.H. Halton, and J.M. Hammersley, "The Shortest Path through Many Points", Proceedings of the Cambridge Philosophical Society, vol. 55, pp. 299-327, 1959.

Brandimarte, P,, "Routing and Scheduling in a Flexible Job Shop by Tabu Search," Dipartimento di Sistemi di Produzione, Politecnico di Torino, Torino, Italy, 1992.

Conway, R.W., W.L. Maxwell, and L.W. Miller, *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1970.

de Werra, D. and A. Hertz, "Tabu Search Techniques: A Tutorial and an Application to Neural Networks", *OR Spectrum*, vol. 11, pp. 131-141, 1989.

Dell'Amico, M. and M. Trubian, "Applying Tabu-Search to the Job-Shop Scheduling Problem," to appear *Annals of Operations Research*, 1992.

Evans, J., "Structural Analysis of Local Search Heuristics in Combinatorial Optimization," *Computers and Operations Research*, vol. 14, no. 6, pp. 465-477, 1987.

Feo, T.F. and M. Resende, "A Probabilistic Heuristic for a Computationally Difficult Set Covering Problem", *Operations Research Letters*, vol. 8, pp. 67-71, 1989.

Fiechter, C-N. "A Parallel Tabu Search Algorithm for Large Traveling Salesman Problems", Research Report ORWP 90/1,Ecole Polytechnique Federale de Lausanne, Departement de Mathematiques, February, 1990.

French, S., *Sequencing and Scheduling*, Ellis Horwood Limited, 1982.

Gendreau, M., A. Hertz, and G. Laporte, "A Tabu Search Heuristic for the Vehicle Routing Problem," Centre de recherche sur les transports, Universite de Montreal, publication #777, June 1991.

Gendreau, M., A. Hertz, and G. Laporte, "New Insertion and Post-Optimization Procedures for the Traveling Salesman Problem," Centre de recherche sur les transports, Universite de Montreal, CRT-708, 1990.

Glover, F., "Heuristics for Integer Programming Using Surrogate Constraints", *Decision Sciences*, vol 8, no. 1, pp. 156-166, 1977.

Glover, F.,"Future Paths for Integer Programming and Links to Artificial Intelligence", *Computers and Operations Research*, vol. 13, no. 5, pp. 533-549, 1986.

Glover, F., "Tabu Search - Part I", *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190-206, Summer 1989.

Glover, F., "Tabu Search - Part II", *ORSA Journal on Computing*, vol. 2, no. 1, pp. 4-32, Winter 1990a.

Glover, F., "Tabu Search: A Tutorial", Interfaces, vol. 20, no. 4, pp. 74-94, July - August 1990b.

Glover, F., "Multilevel Tabu Search and Embedded Search Neighborhoods for the Traveling Salesman Problem," Graduate School of Business and Administration, University of Colorado at Boulder, June 1991.

Glover, F. and M. Laguna, "Tabu Search", University of Colorado at Boulder, 1992.

Glover, F., E. Taillard, and D de Werra, "A User's Guide to Tabu Search," Graduate School of Business and Administration, University of Colorado at Boulder, November 1991.

Hertz, A. and D. de Werra, "The Tabu Search Metaheuristic: How we use it", Research Report ORWP 88/13, Ecole Polytechnique Federale de Lausanne, Departement de Mathematiques, March 1989, to appear in *Annals of Mathematics and Artificial Intelligence.*

Johnson, D., "Local Optimization and the Traveling Salesman Problem," *Proceedings of the 17th Annual Colloquim on Automata, Languages and Programming*, Springer-Verlag, pp. 446-461, 1990.

Knox, J. and F. Glover, "Tabu Search, An Effective Heuristic for Combinatorial Optimization Problems", Center for Applied Artificial Intelligence, University of Colorado, Boulder, Colorado, March, 1988.

Knox, J. and F. Glover, "Comparative Testing of Traveling Salesman Heuristics Derived form Tabu Search, Genetic Algorithms, and Simulated Annealing", Center for Applied Artificial Intelligence, University of Colorado, Boulder, Colorado, September, 1989.

Laguna, M., J. W.Barnes and F. Glover, "Tabu Search Methods for a Single Machine Scheduling Problem," *Journal of Intelligent Manufacturing*, vol. 2, pp. 63-73, 1991.

Laguna, M., and J. L. Gonzalez-Velarde, "A Search Heuristic for Just-in-Time Scheduling in Parallel Machines", *Journal of Intelligent Manufacturing*, vol. 2, pp. 253-260, 1991.

Laguna, M., J. P. Kelly, J. L. Gonzalez-Velarde, and F. Glover, "Tabu Search for the Multilevel Generalized Assignment Problem," Graduate School of Business and Administration, University of Colorado at Boulder, November 1991.

Laguna, M., "Tabu Search Primer," Graduate School of Business and Administration, University of Colorado at Boulder, March 1992.

Laguna, M., J. W. Barnes and F. Glover, "Scheduling Jobs with Linear Delay Penalties and Sequence Dependent Setup Costs and Times Using Tabu Search", *Applied Intelligence*, in press, 1992.

Laguna, M. and F. Glover, "Integrating Target Analysis and Tabu Search for Improved Scheduling Systems", *Experts Systems with Applications: An International Journal*, in press, 1992.

Lam, J., "An Efficient Simulated Annealing Schedule," PH.D. Dissertation, Report 8818, Department of Computer Science, Yale University, September 1988.

Lawler, E.L., J.K. Lenstra, A. Rinnooy Kan, and D.B. Shmoys, "Sequencing and Scheduling: Algorithms and Complexity", Report BS-R8909, Centre for Mathematics and Computer Science, Amsterdam, Netherlands, 1989.

Lawler, E.L., J.K. Lenstra, A. Rinnooy Kan, and D.B. Shmoys, *The traveling Salesman Problem, A Guided Tour of Combinatorial Optimization*, Wiley-Interscience Pub., 1990.

Lawrence, S., Resource Constrained Project Scheduling : An Experimental Investigation of Heuristic Scheduling Techniques, GSIA, Carnegie Mellon University, 1984.

Lin, S. and B. Kernighan, "An Effective Heuristic Algorithm for the Traveling Salesman Problem," *Operations Research*, vol. 21, pp. 498-516, 1973.

Malek, M., M. Guruswamy, H. Owens, and M. Pandya, "Serial and Parallel Search Techniques for Traveling Salesman Problem", *Annals of OR: Linkages with Artificial Intelligence*, vol. 21, pp. 59-84, 1989a.

Malek, M., M. Guruswamy, H. Owens, and M. Pandya, "A Hybrid Algorithm Technique," Report TR-89-06, Department of Computer Sciences, The University of Texas at Austin, Austin,Texas, March 1989b

Matsuo, H., C. Suh, and R. Sullivan, "A Controlled Search Simulated Annealing Method for the General Jobshop Scheduling Problem," Working Paper 03-44-88, Graduate School of Business, University of Texas at Austin.

Morin, T.L. and R.E. Marsten, "Branch and Bound Strategies for Dynamic Programming", *Operations Research*, vol. 24, no. 4, pp. 611-627, 1976.

Muhlenbein, H., "Parallel Genetic Algorithms and Combinatorial Optimization," to appear in *SIAM Journal on Optimization*.

Muth, J. and G. Thompson, *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, N.J., 1963.

Nawaz, M., E. Enscore, and I. Ham, "A Heuristic Algorithm for the m-Machine, n-Job Flow Shop Sequencing Problem," *OMEGA*, vol. 11, no. 1, 1983.

Osman, I. H., "Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem," to appear in *Annals of Operations Research*, 1992.

Padberg, M. and G. Rinaldi, "A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems," SIAM Review, vol. 33, no. 1, pp. 60-100, 1989.

Roy, B. and B. Sussman, "Les problemes d'ordonnancements avec contraintes disjonctives," Notes DS n 9 bis, SEMA, Paris, 1964.

Semet, F. and E. Taillard, "Solving Real-Life Vehicle Routing Problems Efficiently Using Tabu Search," École Polytechnique Fédérale de Lausanne, ORWP 91/03, April 1991.

Stewart, W.R., Jr., "Accelerated Branch Exchange Heuristics for Symmetric Traveling Salesman Problems", *Networks*, vol. 17, pp. 423-437, 1987.

Taillard, E., "Parallel Taboo Search Technique for the Job Shop Scheduling Problem", Research Report ORWP 89/11, Ecole Polytechnique Federale de Lausanne, Departement de Mathematiques, July 1989.

Troyon, M., "Quelques Heuristiques et Resultats Asymptotiques por trois Problemes d'Optimisation Combinatoire," These No. 754, Ecole Polytechnique Federale de Lausanne, Lausanne, Switzerland, 1988.

Tsubakitani, S. and J. R. Evans, "Optimizing Tabu List Size for the Traveling Salesman Problem," College of Business Administration, University of Cincinnati, June 1991.

Ulder, N., E. Aarts, H. Bandelt, P. van Laarhoven, and E. Pesch, "Genetic Local Search Algorithms for the Traveling Salesman Problem," *Lecture Notes in Computer Science*, Vol. 496, Springer, Berlin, pp. 109-116, 1991.

van Laarhoven, P., E. Aarts, and J. Lenstra, "Job Shop Scheduling by Simulated Annealing", Report OS-R8809, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, 1988.

Whitley, D., "Solving Generic Scheduling Problems Using Genetic Algorithms," Technical Report: CIAI-TR-89-04, Colorado Institute of Artificial Intelligence, Boulder, Colorado, 1989.

Widmer, M. and A. Hertz, "A New Method for the Flow Sequencing Problem", *European Journal of Operations Research*, vol. 41, pp. 186-193, 1989.

Widmer, M., "Job Shop Scheduling with Tooling Constraints: a Tabu Search Approach," *Journal of the Operational Research Society*, vol. 24, no. 1, pp. 75-82, 1991.

Woodruff, D. L. and M. L. Spearman, "Sequencing and Batching for Two Classes of Jobs with Deadlines and Setup Times," *Journal of the Production and Operations Management Society*, in press, 1992.

Woodruff, D. L. and E. Zemel, "Hashing Vectors for Tabu Search," *Annals of Operations Research*, in press, 1992.

Measuring the Quality of Manufacturing Schedules

KEVIN GARY[1], REHA UZSOY[2], STEPHEN P. SMITH[3], and KARL KEMPF[3]

[1] *Global Associates Ltd., Arlington, VA*

[2] *School of Industrial Engineering, 1287 Grissom Hall, Purdue University,West Lafayette, IN 47907-1287, uzsoy@ecn.purdue.edu*

[3] *Knowledge Applications Laboratory, Intel Corporation, Chandler, AZ 85226*

Abstract

The understanding of what constitutes a "good" production schedule is central to the development and evaluation of automated scheduling systems and their implementation in real-world factories. In this paper we describe a number of different aspects of this problem and show that it is in fact a very complex problem in its own right. After reviewing work in this area to date, we suggest a new approach which tries to reduce the complexity of the task. We illustrate our approach with an application to evaluating schedules for a semiconductor wafer fabrication facility.

1. Introduction

Although there has been a vast body of work on production scheduling in both the technical literature and industrial practice (Lawler et al.1993; Rodammer and White 1988), the problem of assessing the quality of a given production schedule does not seem to have been emphasized to date. However, a clear understanding of how to assess the quality of a schedule (i.e., what constitutes a "good" schedule) is critical to the successful implementation of scheduling systems in real-world manufacturing environments. In this paper, we present a number of different aspects of the problem of schedule quality assessment and describe efforts to address these to date. Finally, we suggest a new approach which tries to combine approaches so as to make the problem of schedule assessment less complex and provide more meaningful information.

1.1 What is a Schedule?

The most general definition of a scheduling problem is that of assigning scarce resources to competing activities over time in order to obtain the best possible system performance. In this paper, we are interested in the problem of factory scheduling, where the resources are machines and the competing activities jobs that require processing on the machines. Thus we shall refer to a workpiece or batch of workpieces requiring processing on a number of different machines in a particular order as a *job*. The processing performed on a job at a particular machine will be referred to as a *step*. Hence each job requires a number of steps, whose order is often specified by the technology in use or the part geometry. A number of different machines may be capable of performing a particular step on a particular job. While there may be multiple resources such as operators and tooling to be scheduled in a manufacturing environment, we shall restrict ourselves to machines as the main resource we discuss in this paper.

Given this definition of the scheduling problem, we shall consider a schedule to consist of a set of start times and machine assignments for each step of each job to be scheduled. In a manufacturing facility, the input to a scheduling system is the current location of jobs in the system and the state of the machines to process them (available or not). The output from the system will be the set of job/machine/time assignments for a given time horizon.

A schedule may be used on the shop floor in a number of different ways. Generally a schedule is intended to produce certain patterns of behavior in the manufacturing facility for which it was generated. We shall refer to a schedule as a *predictive schedule* when it is released to the shop floor at the beginning of the planning period with the intention of guiding the behavior of the system over the time horizon. How strictly a predictive schedule is adhered to will vary from one manufacturing environment to another. At one extreme, a schedule can be viewed as advice to the personnel on the manufacturing floor which they will follow to some degree. At the other extreme, a schedule may carry operational semantics which directly drive an automated manufacturing system. In either case, there may be substantial deviations from the predictive schedule over the course of its execution due to unforeseen disruptions such as machine breakdown or shop floor personnel overriding the predictive schedule. An actual set of job/machine/time assignments realized on the shop floor will be referred to as a *historical schedule*. We can think of the predictive schedule as being the schedule as designed, and the historical schedule as the schedule as executed. Clearly both types of schedules are of interest from the point of view of schedule quality assessment.

1.2 How Good is a Schedule?

Whether we are dealing with a predictive or a historical schedule, the question "what makes a good schedule good?" is a valid one. The first condition that any schedule must satisfy is feasibility - it should not violate any physical constraints in the manufacturing system in which it is to be executed. In other words, its execution over the specified time horizon should be physically possible. It must put jobs through steps in the appropriate order specified by the product routings. It must assign steps to machines capable of running those steps. These types of constraints must be satisfied for the schedule to be considered feasible.

A second condition concerns acceptability. An acceptable schedule is one that cannot be improved by trivial changes. For example, it may be feasible to run Step A on Machine 1 or Machine 2, but it may be preferable to use Machine 1 when possible due to its being capable of meeting tighter tolerances. A schedule which runs Step A on Machine 2 while Machine 1 sits idle can clearly be improved by using Machine 1.

In this paper we shall assume that a schedule that is not both feasible and acceptable can be dismissed at once. Once a feasible, acceptable schedule is available, the problem of deciding how good that schedule is can be addressed. This question can be approached from a number of different measurement perspectives including: static versus dynamic, absolute versus relative, schedule versus state, and individual versus group. These ideas are discussed briefly before addressing metrics at a more concrete level. For the rest of this paper we will use the term "to measure a schedule" to denote the act of assessing the quality of a given schedule.

Static measurement versus dynamic measurement. Static measurement of a schedule involves measuring a schedule independently of the execution environment. For a predictive schedule, the question is how good the result would be if the schedule was executed exactly as specified. For a historical schedule, we wish to determine how good the outcome of the execution of that schedule was, independently of the originally intended course of action expressed in a predictive schedule.

Dynamic measurement of a schedule is more difficult. In addition to the static quality of the schedule itself, we consider how robust the schedule is in the face of disruptions, such as machine breakdowns, that occur during execution. An attempt to quantify this aspect of a schedule would require a description of the types of disruptions as well as their probabilities of occurrence and a specification of the capabilities of the execution agent that will react to these disruptions. Without knowing how the agent deals with unexpected events, the true impact of the events cannot be measured. The problems of assessing the quality of a schedule from the dynamic perspective is of considerable practical significance. However, due to the difficulty of constructing meaningful metrics for this type of measurement, we restrict our attention to static measurement of schedules in this paper.

Absolute measurement versus relative measurement. An absolute measurement of schedule quality consists of taking a particular schedule on its own and deciding how good it is. This requires some criteria or benchmarks against which to measure - some abstract definition of schedule quality. This is difficult even if we are interested in optimizing with respect to a single metric. Ideally we would like to compare the result of the schedule to the optimal schedule, but since most factory scheduling problems are NP-hard (Garey and Johnson 1979),

this is computationally impractical. An alternative is to use upper and lower bounds on the metrics of interest. For example, it is possible to compute how far a job could have gotten through the shop under perfect circumstances over the time horizon and to compare this to the performance of the job in the schedule in question. However, it is often difficult to compute bounds that are tight enough to provide meaningful information.

A relative measurement assumes that two or more schedules for the same initial factory state are available, and the task is to decide which is better. While this may appear to be a more tractable problem given that all the candidates are available for detailed inspection, two difficulties remain. In most manufacturing environments one is interested in a number of different metrics of schedule performance. Hence it is extremely likely that a given schedule will do better than another for some metrics and worse for others, making it difficult to pick a clear winner without addressing the issue of how to trade the different metrics off against each other. We shall return to this issue of evaluating schedule quality in the face of multiple, possibly conflicting metrics later in this paper. Another issue is that all the schedules under consideration may be poor by absolute standards and this will go undetected during the comparisons.

If the scheduling system under consideration is being evaluated in the context of an operating manufacturing facility, historical information can be used. For absolute measurement, the historical average or best-case performance of the factory can be used, perhaps as the lower bound in conjunction with the upper bounds mentioned earlier. In the case of relative measurement, average historical performance can be used as a benchmark for comparison. Another possibility is to measure the performance of a schedule against trends of historical data. This would allow the effects of continual improvement in the scheduling system to be observed over a period of time.

The use of historical data highlights the idea that real data can be used as a sanity check and automated systems should (at least) provide improvement on existing systems. However, care must be taken to update historical data since most manufacturing systems undergo constant change.

Schedule measurements versus state measurements. In measuring the quality of a given schedule, we are often interested in the activities that took place over the time horizon. Typical questions concern the amount of work completed by a given machine or the progress

towards completion made by a job over the time horizon. These types of measurements which relate to the scheduled activities we shall refer to as *schedule measurements*. However, schedule measurements alone are often insufficient. For example, a schedule for a given machine may perform very well for that machine over that time period, but may leave it in an impossible position at the end of the time horizon. In this type of situation it is desirable to evaluate the state the schedule leaves the factory in. One set of such measurements are those relating to the location of Work-In-Progress (WIP) inventory relative to available capacity. One would expect these measurements to be position based, rather than activity-based, and to describe the distribution of work over time or across the different work areas in the manufacturing system being scheduled. These latter types of measurements we shall refer to as *state measurements*.

Measuring individual schedules versus groups of schedules. Another aspect of the measurement issue is whether we are measuring an individual schedule or a group of schedules. An obvious reason to measure an individual schedule is to gauge its individual performance. For a predictive schedule, the result may determine whether or not it will be implemented. For a historical schedule, the result may be used to evaluate the performance of shop-floor personnel over the time horizon.

Probably the most common reason for measuring groups of schedules is to evaluate the strategy or algorithm being used to develop them. A strategy might be applied to a set of beginning states which cover the extremes found in practice to produce a group of schedules and ending states. Measurement of the group of schedules can be used to draw conclusions about the strategies producing them.

1.3 Selection of Metrics to Apply

The issue of what metrics to use to assess the quality of a schedule is far from trivial. Answering this question is equivalent to describing the kind of behavior we want the scheduling system to induce in the manufacturing system, which in turn is equivalent to deciding what the goals of factory management should be. Given our definition of schedules as sets of job/machine/time assignments, it seems natural to restrict our attention to metrics that can be calculated based on the schedule itself, starting from these assignments. Restricting our attention to static,

relative measurement further narrows the alternatives. However, there are a number of factors that make the choice of a meaningful set of metrics from among the wide range of mathematically and intuitively plausible ones difficult.

Organizational Goals. A major problem is that the different organizational units affected by the schedule have different, often conflicting goals and as such have substantially different expectations from a schedule. A marketing department will often look at a schedule from the point of view of orders being delivered to the customer. A manufacturing department, on the other hand, may be under pressure to reduce costs. Thus the marketing department will look for a schedule that has good due date performance, while the manufacturing department will prefer a schedule with high machine utilization, few setups, and long production runs. An interesting discussion of this type of problem is given by Harrison et al.(1989). One can argue that this problem can be alleviated by assigning the different manufacturing organizations goals which are coherent and directed towards the overall good of the company as a whole. However, this problem of setting compatible performance measures for different organizational units seems to be a long way from being solved.

The question of what metrics should guide a scheduling system to ensure that the schedules it generates are consistent with the longer-term goals of the company is a difficult one, and does not seem to have been addressed in the literature to date. As one moves up the corporate hierarchy, goals tend to be expressed more and more in aggregate and financial terms, culminating in the very brief corporate mission statement. To complicate the issue further, the time horizons over which decisions are made differ vastly over different levels - from years and months at the corporate level to weeks and days at the plant level and hours and minutes on the shop floor. It is very difficult to reconcile a goal like "maximize profit for this quarter" with the decisions made while scheduling an eight-hour shift on the shop floor. The problem of developing coherent sets of performance measures for all levels of the corporate hierarchy to ensure that all levels are working towards the same set of corporate objectives and not adversely affecting each other seems to be an open one at present (Fisher 1992).

Metric Relationships. The existence of many plausible metrics for evaluating schedule quality raises the question of relationships between the metrics. Some metrics are complementary, an improvement in

one bringing with it an improvement in the other. An example might be minimizing cycle time variance and improving on-time delivery performance. However, many metrics that are potentially conflicting are also of interest in scheduling. Maximizing machine utilization and minimizing WIP serve as an example. Maximizing the utilization of machines can be achieved by assuring that work is always available at each machine by maintaining high inventories, while minimizing the WIP might risk machines being starved at some point in time. Such interactions between metrics further confound the process of measuring schedules.

Aggregation and Segmentation of Metrics. When assessing the quality of a given production schedule, the raw data used to calculate static metrics is set of job/machine/time assignments that constitute the schedule. Examples of these metrics, which we shall refer to as *atomic metrics*, are the time spent in process by a particular lot at a particular machine, or the proportion of the time horizon that a certain machine was busy or down. However, it is apparent that we can use these atomic metrics in a number of different ways to calculate metrics relating to the overall schedule being evaluated. We shall distinguish between two different operations, segmentation and aggregation, that are performed on the atomic metrics.

The operation of *segmentation* consists of specifying a class of scheduling objects (jobs or machines) that form a meaningful unit in terms of schedule evaluation. To illustrate this idea, consider the set of all jobs in the schedule. The atomic metrics available are time spent waiting and in process at each step. Possible segmentations of these atomic metrics are by jobs of the same product, or by jobs of the same priority class (urgent/late/on time, for example). A commonly encountered segmentation is by administrative area - for example, the fabrication and assembly areas of a manufacturing plant might be viewed as segments of the scheduling problem. The underlying idea is that the performance of a given schedule will be evaluated in terms of its impact on the set of scheduling objects specified in the segmentation.

Once the segmentation of the set of scheduling objects, and thus of the associated set of atomic metrics, has been specified the atomic metrics for each segment can be *aggregated* in a number of different ways. The most common method of aggregation is averaging. Thus, one might aggregate a set of atomic metrics by taking their average over a given set of jobs or machines specified in the segmentation; one might take the average of a set of atomic metrics associated with a given object

over the length of the time horizon; or one might do both. There seem to be two main dimensions along which one can aggregate atomic metrics: the set of scheduling objects specified as a meaningful unit for schedule evaluation in the segmentation, and some specified time horizon. Each of these dimensions in turn may imply a hierarchy of aggregation. For example, if we segment the atomic metrics by functional departments on the shop floor, such as drilling, milling and welding, we will aggregate the atomic metrics for each department into an aggregate metric for that entire department. However, the aggregate departmental metrics can, in turn, be aggregated into plant-wide metrics which can then be aggregated into company-wide ones. If we are aggregating over time, the atomic metrics may be aggregated over a time horizon such as an eight-hour shift. These metrics may then be aggregated into daily, weekly, monthly and yearly metrics.

The segmentation and aggregation of metrics is crucial to good schedule quality evaluation. It will also affect the behavior of the manufacturing system, since the way the metrics are structured will determine the criteria by which the performance of shop-floor management will be evaluated, which in turn will affect the decisions made by shop-floor management. The segmentation must be over classes of objects that are meaningful not just in terms of organizational structure, but in terms of the manufacturing system being scheduled. If we segment the problem based on functional departments, as is often the case in industry, we run the risk of that department optimizing its own performance independently of other units in the plant. The way in which metrics are aggregated is closely linked to the problem of establishing meaningful metrics for different levels of the corporate hierarchy discussed above. Ideally, the aggregation of metrics should be accomplished in a way that will ensure the coordination and coherence, in terms of overall corporate goals, of the actions taken in the schedule. The degree of aggregation is an important variable here. If we aggregate too little, we face a mass of atomic data from which it is difficult to draw any conclusion about the schedule as a whole. If we aggregate too much, we obtain metrics that are very difficult to relate to actual events on the shop floor or characteristics of the schedule. These highly aggregate metrics can often be misleading when used indiscriminately. A good example is a plant-wide average of machine utilization. Trying to maximize this metric will lead to attempts to reach 100% utilization on all machines in the plant, resulting in a dramatic increase in WIP levels and shop-floor congestion. Looking at average utilization on bottleneck machines, for instance, makes a lot more

sense since we are maximizing utilization where it matters, instead of indiscriminately across the whole plant. The challenge is to select a segmentation and an aggregation that provide the most meaningful information to the decision maker who is trying to evaluate the schedule.

The above discussion should highlight the fact that the basic problem of measuring the quality of a schedule is actually a very complicated question that can be addressed from a number of different perspectives. In the rest of this paper we shall focus on static, relative measurements and include both state and schedule related metrics. We shall examine the performance of individual schedules, as well as that of two families of schedules.

2. Previous Related Work

The problem of evaluating schedule quality has been addressed in a number of different ways in the literature to date. By far the great majority of the optimization based work in the literature has assumed that a single metric is applicable across the entire problem. A detailed survey of this body of work can be found in Lawler et al.(1993). This approach is required for the mathematical formulations of scheduling problems as optimization problems to be meaningful. The problem of scheduling in the face of multiple, conflicting objectives has been examined very little and for very simple systems. A survey of research on bicriterion scheduling problems can be found in Dileepan and Sen(1988). There seem to have been three major approaches used in this literature, which we will briefly describe below.

One body of research has addressed scheduling problems with primary and secondary criteria. In this approach, the problem is to minimize the primary metric as far as possible while keeping the other within some predefined range. This is accomplished by constraining the secondary objective to be within the desired range, thus converting the secondary metric into a constraint. The work of Smith(1956) on minimizing total completion time subject to no tardy jobs is the earliest work in this area. This approach has since been followed for a number of different problems by a number of researchers(Miyazaki 1981; Sen and Gupta 1983; Sen et al. 1988; Bansal 1980; Chand and Schneeberger 1986, 1988; Potts and Van Wassenhove 1983). This research, however, has remained limited to two criteria, one primary and one secondary. In addition, no system more complex than a single machine with all jobs available si-

multaneously has been addressed to date. In this approach, the tradeoff between the two criteria is explicitly specified by the definition of primary and secondary metrics and the range for the secondary metric.

Another approach that has been followed uses the ideas of dominated and efficient solutions. Given a number of metrics of interest f_i, i = 1,..,n to be minimized, let f_{ij} denote the value of metric i obtained from schedule j. A schedule j is said to dominate schedule k if $f_{ij} < f_{ik}$ for all i = 1,..,n. A solution that is not dominated in this way by any other solution is called a non-dominated or efficient solution. Given a scheduling problem with multiple conflicting objectives, the approach is to generate the entire set of efficient solutions and let the decision-maker decide which solution is preferable. In this approach the decision as to how the different metrics are to be traded off against each other is left to the human decision-maker. This approach has been followed by van Wassenhove and Gelders(1980) and Nelson, Sarin and Daniels(1986) among others.

One can also combine different metrics into a single metric by taking as a surrogate metric a weighted sum of the original metrics of interest. In this approach, the weights capture the tradeoffs between the different metrics which are left to the decision-maker in the previous approach. Sen et al.(1988) have followed this approach to minimize linear combinations of flow time and range of lateness on a single machine. The difficulty of this approach is that since the weights determine the tradeoffs between the different metrics, the determination of the weights is a non-trivial problem. Other examples of this approach are the development of dispatching rules based on a weighted combination of different job and machine attributes affecting different metrics (Bhaskaran and Pinedo 1991), referred to as composite dispatching rules in the literature.

Approaches in the artificial intelligence literature have, in general, followed a similar approach. Fox and Smith(1984) approach the problem of multiple objectives in job shop scheduling by setting aspiration levels for each metric of interest, thus effectively turning each metric into a constraint. One then attempts to find a solution that is feasible with regard to all constraints. In the event that this is not possible, the constraint corresponding to a given metric will be relaxed until a feasible solution is finally found. The difficulty here is in determining in what order the constraints should be relaxed, and by how much. This problem is again equivalent to that of determining the tradeoffs between metrics. Other researchers, like Elleby et al.(1989), have addressed this

problem by having the user interactively specify the relaxation of the constraints.

All in all, we see that the problem of scheduling in the face of multiple conflicting metrics has been addressed in a variety of ways, all of which reduce to a different way to represent the tradeoffs between the different metrics of interest. In all these approaches, however, it is interesting to note that the metrics of interest are applied across the entire problem, that is, across all objects in the system to which their application makes sense.

In industrial environments, a different approach to schedule evaluation seems to have been followed. In most cases the manufacturing facility is divided into a number of smaller entities, each of which is an independent organizational entity. An example of this is a semiconductor wafer fabrication facility (a fab), where the overall fab is divided into departments by machine function, such as lithography, etching, and diffusion. The performance of each of these areas is assessed based on a number of different metrics. The management of each area then tries to achieve the best possible performance in terms of these different metrics.

Shop-floor management tends to approach the problem of scheduling within a given workcenter or department from the perspective that not all parts of the problem are equally important. Generally, their experience has shown that if a certain machine group or job class can be satisfactorily scheduled, the rest of the schedule will fall into place relatively easily. Thus they tend to focus on special classes of scheduling objects, such as important jobs, bottleneck machines, or operations with a history of quality problems. As a result, within a given class of scheduling objects such as resources or jobs, a number of subclasses are created which are treated differently. For example, within the class of jobs, there may be subclasses corresponding to "hot" jobs, which must be rushed to completion as fast as possible; "standard" jobs, where the main concern is to meet due dates that are far in the future relative to the "hot" jobs; and "low priority" jobs, where the primary goal is to minimize the cost of manufacturing. For machine resources, one might define the class of constraining or "bottleneck" machines; the "high-cost" machines which must be used economically; and the non-constraining resources where minimization of WIP may be the metric of interest.

Although this approach seems to be intuitive, it is not without its own disadvantages. Dividing the facility into different areas by function, for example, often results in different areas optimizing their own schedules at the expense of leaving the next area they feed work to in an im-

possible position. The idea that some parts of the scheduling problem are more important than others is certainly sound, and has been discussed in the literature for a number of years, most vocally by Goldratt in his work on the Theory of Constraints (Goldratt 1986). However, it would seem that in order for this approach to be successful the different subproblems corresponding to subclasses of scheduling objects and the metrics related to those subclasses must be defined "correctly", in a way that does indeed capture the structure of the underlying scheduling problem and the goals of the company.

In the next section, we suggest an approach that tries to capture the advantages of the decomposition-based approach of shop-floor management while trying to eliminate some of the disadvantages discussed above. We decompose the scheduling problem into a number of subproblems based on the different subclasses of scheduling objects, the metrics appropriate for each subclass and the relative importance of each subclass. In order to avoid suboptimization we try to define meaningful ways of aggregating metrics and include measures of system state before and after the execution of a schedule. Our approach is described in detail in the following section.

3. Decomposition-Based Schedule Measurement

In this section of the paper we shall describe the idea of decomposition-based schedule quality assessment that tries to provide a new approach to schedule quality assessment. We shall first give a broad outline of the approach, and then illustrate its application to the problem of measuring schedules developed for a semiconductor wafer fabrication facility.

As mentioned in the previous section, the idea that not all parts of a scheduling problem are equally important seems to be common among practitioners, and is steadily gaining acceptance in academic circles. Our approach to schedule quality assessment is based on this insight. We assume that we can decompose a scheduling problem into a number of subproblems relating to different classes of scheduling objects, such as jobs and machines. Each class of scheduling objects so defined will have a metric associated with it. Rather than apply a basic set of metrics across the entire set of scheduling objects, metrics appropriate to each subproblem are selected and applied only to the subset of scheduling objects defining that subproblem. An example of such a de-

composition would be to look at hot lots, where the goal is to complete them as fast as possible, and bottleneck machines, where the goal is to maximize throughput. Rather than look at time in system and throughput over all jobs and machines respectively, we would calculate time in system for hot jobs, while calculating the throughput only for the set of bottleneck machines. Thus a schedule will be evaluated on the basis of the set of metrics defined for each of the subproblems. It is natural to look at important entities, such as hot lots, in detail while looking at other, less critical objects in a more aggregate way. Hence the subproblems upon which the metrics for schedule evaluation are defined may differ considerably in level of aggregation.

Thus, we specify a set of subproblems of the original scheduling problem. Each subproblem is defined by a set of scheduling objects and a metric to be calculated over that set of objects. In order to measure a schedule, we calculate the metrics for each subproblem and use these values to compare different schedules. Recall that we have limited ourselves to static, relative measurements of schedules over the same time horizon. Given that we are defining metrics over a number of different subproblems which may represent different levels of segmentation, the success of this approach clearly depends on the choice of subproblems and the metrics associated with the subproblems. The decomposition should capture the critical components of the system in detail, while the segmentations should be meaningful in terms of the manufacturing system under study. In addition, note that the problem of how to trade off the different metrics against each other is still open. The metrics we have chosen for the different subproblems may conflict because of the nature of the subproblems, the nature of the metrics, or both. However, we feel that the decomposition-based approach will result in metrics that are more meaningful to the decision maker comparing schedules, since they relate directly to the critical portions of the system. In addition, rather than calculating all metrics over all subproblems, we calculate only the relevant metrics for each, which ought to result in a smaller set of metrics to be considered. The final decision as to tradeoffs between different metrics can then be addressed by the decision-maker.

To illustrate this approach, we shall apply these ideas to the problem of evaluating schedules for a semiconductor wafer fab. Our objective is to compare two different scheduling strategies for the fab, examining the schedules from the static, relative perspectives using both state and schedule-based metrics. We first briefly describe the scheduling en-

vironment, then present the set of metrics to be used, and finally apply these metrics to compare the two rival strategies.

3.1 Fab Scheduling Environment

Wafer fabrication is the most complex and capital-intensive stage of the semiconductor manufacturing process. It involves the processing of wafers of silicon or gallium arsenide in order to build up the layers and patterns of metal and wafer material required to produce the desired circuitry. The number of unit operations (steps) can be well into the hundreds for a complex circuit such as a microprocessor. Many of these operations must be performed in a clean-room environment to prevent particulate contamination of the wafers. The wafers move through the facility in standard lots (a word we will now use interchangeably with "job"), based on the size of a standard container used to transport wafers. A basic operation sequence of operations consisting of cleaning, deposition, lithography, etching, ion implantation and inspection is repeated for each of the multiple layers of circuitry on the wafer. This results in lots requiring processing at the same workcenter several times, giving the fab the aspect of a reentrant flow shop (Graves et al.1983). Detailed descriptions of the semiconductor manufacturing process can be found in specialized texts such as Sze(1988).

3.2 Basic Metrics

Given this basic structure of the manufacturing environment, the atomic metrics we are interested in can be classified into three groups: lot metrics, resource (machine) metrics and step metrics. Lot metrics include busy time for a lot (i.e., how much time the lot spent actually being worked on), the "critical ratio" of the lot (i.e., the theoretical cycle time remaining in process divided by the time till the lot is due), and number of steps completed (moved) per lot. Step metrics include Work-in-Progress at each step at the beginning and end of the planning period (denoted by BOS- WIP and EOS-WIP respectively) and the number of lots processed at each step over the planning period. Note that the WIP-related metrics are state-related, rather than schedule-related metrics. Machine metrics include machine utilization, availability, and throughput. A definition of each of these atomic metrics is given in Table 1. The

planning period for this study was fixed to be twelve hours, corresponding to the length of a production shift in the fab under study.

3.3 Segmentation/Aggregation of Metrics

As discussed above, atomic metrics may be applied to various segmentations of scheduling objects: all lots, some selected lots, some sequence of steps, etc. We now discuss some important domain-dependent aspects of segmentation for the fab scheduling problem. For aggregating these segmented metrics, we will not go beyond "standard" data summation techniques, such as the minimum, maximum and average of a set of values. A portion of the manufacturing process is shown schematically in Figure 1. Steps are represented by nodes, and the label of the node represents the type of machinery required to process the step.

Lot Metrics:	
BOS-CR	Beginning of Schedule Critical Ratio
EOS-CR	End of Schedule Critical Ratio
SC	Steps Completed. Number of steps performed on hot lots over the schedule horizon
PU	Percent Utilization. Percentage of time in the schedule horizon the lot was being worked on, as opposed to waiting, in transport, etc.
Resource Metrics:	
LC	Lots Completed. Number of lots processed by the resource over the schedule horizon
PU	Percent Utilization. Percentage of time in schedule horizon resource was processing lots.
Step Metrics:	
BOS-WIP	Beginning of Schedule WIP. The number of lots waiting for processing at this step at the beginning of the schedule horizon.
EOS-WIP	End of Schedule WIP. The number of lots waiting for processing at this step at the end of the schedule horizon.
LC	Lots Completed. Number of lots processed through the step over the schedule horizon.

Table 1: Definition of Atomic Metrics

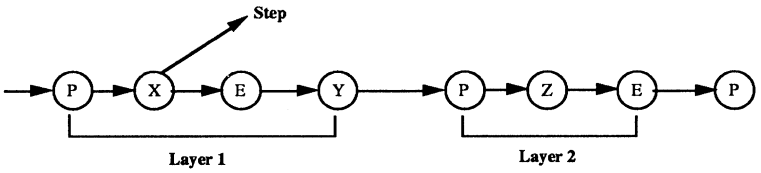


Figure 1. Example Product Flow for Problem Segmentation

Step/Machine Groups. In the wafer fabrication process, a number of steps may require the same resource, and a given resource may be capable of performing a number of different steps. In Figure 1, the two steps represented by the nodes labelled with a "P" are processed on the same set of machines. Hence the set of steps that compete for capacity at a certain set of machines becomes an important aspect of the scheduling problem, since the structure of this set defines the choices available for the scheduler to make. We can specify this set as a set of step-machine pairs which denote whether or not that particular step can be carried out on that machine. The set of all such step-machine pairs which have capacity-related interactions will be referred to as a *step-machine group*. Thus a step-machine group will contain all steps and machines whose scheduling decisions affect each other. In order to determine the step-machine group, we partition the set of steps into maximal subsets such that steps in the same set transitively share a machine on which they can be performed.

Step-machine groups may be "complete", where any step in the group may use any of machine in the group, or they can be "partial". In a partial step-machine group, all machines are not capable of processing all steps. An example of a partial step/machine group is given in Figure 2. The step/machine group is represented by a matrix whose ij'th entry is an X if step i can be processed on machine j, and blank otherwise.

	Machine 1	Machine 2	Machine 3	Machine 4
Step 1	X	X		
Step 2	X	X	X	
Step 3		X	X	X
Step 4			X	X

Figure 2: A Partial Step/Machine Group

Given their central role in determining problem complexity, step/machine groups form a logical segmentation of the problem for mea-

surement purposes. It is clear that some step/machine groups will be capacity limited, thus becoming bottlenecks in the production process and making them more important than others. Thus we shall segment the atomic metrics listed above over step/machine groups. For instance, the number of lots moved out of all steps in a group gives a relative measure of line flow when compared among groups. Intra-group comparisons of each step's lot completions give tradeoffs made between the steps competing for the same resources. Figure 3 shows the diagram of a simple complete step/machine group with two steps and two machines and the various metrics computed from entities in the group.

	Machine A	Machine B	
Step 1	Lots (1,A)	Lots (1,B)	Lots at Step 1
Step 2	Lots (2,A)	Lots (2, B)	Lots at Step 2
	Lots at Machine A	Lots at Machine B	All lots at group

Figure 3: Example Step/Machine Group Segmentation

Bottleneck Step/Machine Groups. Two particular step/machine groups stand out as bottlenecks in the fab under study: the "E" group and the "P" group, whore precise nature is withheld for proprietary reasons. Simple capacity calculations show that the two steps in the "E" group are the flow limiting steps of the process. The "P" group stands out since its steps are close to being flow limiting steps, but the steps in this group are also spaced out throughout the whole re- entrant process flow.

Layers in the Process Flow. Because of the semantics of the process, it is natural to identify a step with the "P" step which precedes it in the process flow. One can say that the process flow is partitioned into a sequence of "layers" which correspond to the process steps necessary to fabricate one of the physical layers of circuitry on the microelectronic device under manufacture. Hence an important segmentation is the movement of material through these layers. The number of lot completions over all the steps in a layer, compared to other layers, shows how much the schedule "concentrated" on that layer. An example of a layer is shown in Figure 1.

Hot Lots. For various reasons which may or may not be schedule related, certain lots in the fab are designated as "hot" lots with high priority that are to be expedited through the fab. Every effort should be made to keep hot lots moving through their remaining process steps, as

long as the movement of other lots is not affected to "too great" an extent by doing so.

Given the discussion above, we can now summarize the decomposition we shall use for assessing the quality of the fab schedules as follows:

(1) Hot lots, where we wish to minimize time in system;
(2) Bottleneck step/machine groups, where we wish to maximize throughput;
(3) Layers of the process flow, where we wish to have high, even throughput across layers.

3.4 The Strategies

The two strategies whose performance we wish to compare are implemented using the strategy language of an AI-based predictive production scheduling system (Kempf et al. 1991). The approach of the scheduling system is based on "importance ordered" predictive scheduling. In this technique, before a predictive schedule is started, an empty Gantt chart for the entire fab is created and the current state of the fab laid down upon it. Using this initial Gantt chart as a starting point, the scheduler constructs its schedule starting with the most important subproblems. The function of the strategy is to define the order in which specific subproblems, corresponding to sets of scheduling objects, should be placed in the schedule.

We shall describe the two scheduling strategies being compared very briefly, since we are interested in the issue of assessing their quality rather than in the details of their operation, omitting many internal details such as how batches are formed and how resources and free periods on resources are chosen for assignment.

Strategy A. This strategy arises from the conventional wisdom that the bottleneck operations in the "E" step/machine group should be where the scheduler concentrates its efforts. The basic logic of this strategy is as follows: If there are starts into the process, schedule them into the Gantt chart, moving them as far through the process as possible. For all the lots in each "P" layer immediately preceding the "E" steps, try to move the lot up to and through the "E" step. Then schedule Hot Lots, and finally other lots.

Strategy B. This strategy is motivated by the observations of floor supervisors, who felt that they had a hard time deciding tradeoffs between the different steps competing for capacity at the "P" step/machine group. The intuition is that a strategy which keeps lots flowing through the "P" steps, which are spaced out over the entire process flow, should also feed the "E" group machines evenly. To maintain an even flow and keep any "P" step from "starving", buffers of WIP lots are maintained at each "P" step.

The detailed logic of this strategy is as follows: If there are lots being started into the fab, schedule them first. Schedule all the lots in the final "P" layer to move out of the process. Schedule lots in the buffer of each "P" step to move through the "P" step so that the WIP buffer is reduced to a minimum target level. This begins at the final "P" layer and works its way forwards in the process flow. Schedule Hot Lots. From among the lots proceeding each "P" step, schedule lots to fill up the "P" step's buffer, so that the buffer is refilled back to a target fill level. Schedule all remaining lots.

3.5 Experiments

To compare the performance of the two strategies using the decomposition-based approach, the strategies were used to generate schedules for the fab at three different levels of loading. The first loading level corresponds to the current state of the fab. The other two scenarios have respectively 10% and 20% more inventory and a correspondingly higher start rate. The results of the comparisons for each class of metrics is described below.

Hot Lots. Table 2 gives the results of the two strategies for hot lots. The numbers are of the form Min/Max/Total for steps and lots completed and Min/Max/Average for all other measures.

	Normal	10%	20%
Strategy A			
PU	8 / 99 / 56	7 / 96 / 68	0 / 88 / 42
SC	0 / 20 / 50	1 / 13 / 54	0 / 19 / 50
BOS-CR	0.2 / 1.0 / 0.4	0.3 / 0.4 / 0.4	0.1 / 0.6 / 0.4
EOS-CR	0.2 / 0.5 / 0.3	0.3 / 0.4 / 0.3	0.3 / 0.8 / 0.4
Strategy B			
PU	35 / 99 / 61	9 / 95 / 48	0 / 88 / 39
SC	0 / 15 / 51	1 / 8 / 47	0 / 19 / 57
BOS-CR	0.2 / 1.0 / 0.4	0.3 / 0.4 / 0.4	0.1 / 0.6 / 0.4
EOS-CR	0.1 / 0.4 / 0.3	0.3 / 0.4 / 0.4	0.3 / 0.8 / 0.4

Table 2: Hot Lot Results

The strongest trend is in the difference between the 10% and 20% WIP levels for both strategies. Percent Utilization declines, while Critical Ratio increases. Note also that the maximum CR increases dramatically while minimum PU goes to zero. This suggests that some hot lots are not being scheduled because of the increased number of lots already placed into the Gantt chart. For instance, lots started into the line are scheduled before hot lots. These lots use machines in the step/machine groups needed by hot lots. The anomalous result that more steps are executed overall in Strategy B although utilization goes down may indicate that the delayed lots are in WIP at steps with longer cycle times. The fact that lots starting into the process require a step with a long cycle time on a machine needed elsewhere in the process supports this conjecture.

"P" Step/Machine Group. Table 3 gives the results of step and machine metrics averaged over all the steps and machine in the "P" step/machine group. For the WIP metrics, the parenthesized number at the end of the EOS totals is the total number of lots in process at one of the steps. Together with the WIP total this gives the amount of work at the steps. Once again the table entries are in the form of minimum/average/maximum.

Strategy B shows better performance by these metrics. Utilization and throughput at the bottlenecks are significantly greater than under Strategy A. In addition, Strategy A shows greater variance in its minimum and maximum values than Strategy B. This is not unexpected since Strategy A emphasizes a small subset of the "P" steps, while the Strategy B emphasizes the class as a whole.

	Normal	10%	20%
Strategy A			
LC (Steps)	4 / 35 / 165	0 / 36 / 160	0 / 40 / 156
LC (Machines)	8 / 13 / 165	1 / 13 / 160	1 / 14 / 156
PU (Machines)	62 / 99 / 84	25 / 100 / 75	19 / 98 / 77
BOS WIP	8 / 24 / 164	10 / 29 / 204	12 / 34 / 237
EOS WIP	1 / 22 / 115 (17)	0 / 34 / 169 (13)	2 / 52 / 242 (15)
Strategy B			
LC (Steps)	4 / 24 / 180	8 / 32 / 203	5 / 28 / 196
LC (Machines)	8 / 16 / 180	10 / 16 / 203	10 / 15 / 196
PU (Machines)	59 / 100 / 89	74 / 100 / 92	69 / 100 / 92
BOS WIP	8 / 24 / 164	10 / 29 / 204	12 / 34 / 237
EOS WIP	1 / 18 / 103 (14)	0 / 26 / 126 (11)	0 / 29 / 201 (12)

Table 3. "P" Step-Machine Group Results

Layer Measures. Critical ratios and WIP levels for the layers in both strategies are similar except for an interesting difference for layers late in the process. Table 4 gives edited metric results for layers occurring towards the end of the process flow. The row data here is given in groups of six, one for each schedule run. These are given as BN, B-10, and B-20, and AN, A-10, and A-20 respectively for Strategy B and A and the corresponding load level. The left hand column indicates the layer by the number of the first step in the layer. The column data represents the five metrics applied to layers. All step metrics are aggregated into compound layer metrics. Again the number of lots in process at EOS over all the steps is noted by the parenthesized number after EOS-WIP.

The interesting result here is the difference in the number of moves between layers i through l. These four layers show a much higher number of moves at each WIP level for Strategy A. Strategy B shows higher move levels in most of the other layers, but most are not very significant. The higher number of moves later in the line for Strategy B may be due to the fact that the strategy emphasizes feeding the "E" steps later in the line. Therefore WIP at steps later in the line gets processed first. Strategy B also tends to emphasize steps later in the line, but does not specifically feed the "E" steps. The result is an increased, but more even, level of movement in the line.

3.6 Comparison of Strategies

Based on the results above, it would seem that Strategy B is preferable to Strategy A for the wafer fab scheduling problem. There is very little difference between the two strategies for hot lots, and Strategy B performs better for throughput on bottleneck step/machine groups, and marginally better for flow through the layers. This example illustrates the fact that the decomposition approach, by focusing attention on particular sets of scheduling objects and on metrics important to that class allows us to focus better on the important comparisons while comparing schedules.

4. Conclusions

In this paper, we have discussed several different aspects of the problem of assessing the quality of manufacturing schedules. We have described a variety of reasons why this problem, which is clearly of great theoretical and practical importance, is difficult to resolve unambiguously, and suggested a new approach. This approach is based on decomposing the scheduling problem into a number of key subproblems and allowing metrics to be defined for each.

To illustrate this method, we have given an example application in the domain of scheduling semiconductor wafer fabrication facilities. We compared two scheduling strategies using key domain subproblems and relevant metrics. Our metrics measured both schedule performance and system state changes for the two strategies over various capacity loads.

	BOS-CR	EOS-CR	LC	BOS-WIP	EOS-WIP (proc.)
Layer i					
AN	0.3 / 1.0 / 0.4	0.3 / 0.7 / 0.4	0 / 36 / 357	0 / 24 / 98	0 / 23 / 71 (25)
A-10	0.3 / 0.5 / 0.4	0.3 / 0.6 / 0.4	0 / 38 / 339	0 / 29 / 94	0 / 38 / 78 (21)
A-20	0.3 / 0.6 / 0.4	0 3 / 0.6 / 0.4	0 / 38 / 367	0 / 34 / 144	0 / 36 / 144 (25)
BN	0.3 / 0.6 / 0.4	0.3 / 0.6 / 0.4	0 / 34 / 249	0 / 24 / 94	0 / 34 / 85 (6)
B-10	0.3 / 0.5 / 0.4	0.3 / 0.6 / 0.4	0 / 32 / 297	0 / 29 / 109	0 / 23 / 71 (18)
B-20	0.3 / 0.6 / 0.4	0.3 / 0.7 / 0.4	0 / 25 / 194	0 / 34 / 144	0 / 28 / 135 (6)
Layer j					
AN	0.2 / 0.6 / 0.4	0.3 / 0.7 / 0.4	0 / 22 / 264	0 / 16 / 85	0 / 15 / 54 (18)
A-10	0.2 / 0.8 / 0.4	0.2 / 1.2 / 0.4	0 / 40 / 316	0 / 20 / 123	0 / 24 / 90 (21)
A-20	0.1 / 0.8 / 0.4	0.2 / 0.9 / 0.4	0 / 38 / 310	0 / 23 / 125	0 / 26 / 109 (23)
BN	0.3 / 0.6 / 0.4	0.3 / 0.7 / 0.4	0 / 25 / 246	0 / 16 / 85	0 / 19 / 61 (20)
B-10	0.2 / 0.8 / 0.4	0.2 / 1.2 / 0.4	0 / 19 / 170	0 / 20 / 123	0 / 26 / 108 (9)
B-20	0.1 / 0.8 / 0.4	0.2 / 0.9 / 0.4	0 / 19 / 132	0 / 23 / 125	0 / 23 / 123 (9)
Layer k					
AN	0.2 / 1.0 / 0.4	0.2 / 2.1 / 0.5	5 / 35 / 223	1 / 16 / 78	0 / 22 / 51 (9)
A-10	-15.4 /18.7/ 0.5	-0.2 / 3.6 / -0.5	1 / 39 / 267	2 / 20 / 99	0 / 34 / 82 (7)
A-20	-2.7 / 3.1 / 0.5	-16.1 / 4.0 / 0.4	5 / 32 / 331	4 / 23 / 125	0 / 37 / 97 (14)
BN	0.2 / 1.0 / 0.4	0.2 / 2.1 / 0.6	8 / 24 / 230	1 / 16 / 78	0 / 12 / 65 (7)
B-10	-15.4 / 18.7 / 0.5	-2.1 / 3.8 / 0.5	5 / 28 / 253	2 / 20 / 99	0 / 25 / 91 (11)
B-20	-2.7 / 3.1 / 0.5	-15.3 / 4.9 / 0.4	3 / 24 / 237	4 / 23 / 125	0 / 20 / 95 (16)
Layer L					
AN	-2.4 / 7.3 / 0.6	-3.9 / 2.7 / 0.4	0 / 51 / 439	0 / 16 / 73	0 / 18 / 66 (20)
A-10	0.2 / 5.2 / 0.8	-37.0 / 9.9 / 0.2	0 / 57 / 352	0 / 21 / 70	0 / 21 / 75 (14)
A-20	-14.0 / 2.9 / 0.3	-2.2 / 7.8 / 0.6	0 / 51 / 459	0 / 23 / 120	0 / 22 / 101 (12)
BN	-2.4 / 7.3 / 0.6	-2.2 / 2.8 / 0.4	0 / 27 / 269	0 / 16 / 73	0 / 15 / 59 (9)
B-10	0.2 / 5.2 / 0.8	-31.1 / 6.0 / 0.2	0 / 20 / 165	0 / 21 / 70	0 / 17 / 64 (4)
B-20	-14.0 / 2.9 / 0.3	-2.0 / 8.3 / 0.6	0 / 38 / 329	0 / 23 / 120	0 / 23 / 93 (16)

Table 4: Layer Results

Clearly, much remains to be done with this approach. First, a more extensive formal description of the decomposition approach should be undertaken and more detailed experiments with its use should be performed. Second, more automatic methods to allow users to define subproblems and key metrics should be implemented. Other directions in which we have been progressing include an explanation system which follows up in a key problem area by highlighting the underlying cause for its corresponding metric score. In addition, we hope that this paper will stimulate research into the various aspects of schedule quality assessment, which seems to have been largely ignored in the past.

Acknowledgments

The research of Reha Uzsoy was performed while a visiting researcher at Intel Corporation in July-August 1992.

References

Bansal, S.P., "Single Machine Scheduling to Minimize Weighted Sum of Completion Times with Secondary Criterion - A Branch and Bound Approach", *European Journal of Operational Research* 5, 177-181, 1980.

Bhaskaran, K., Pinedo, M., "Dispatching", Chapter 83, *Handbook of Industrial Engineering*, G. Salvendy (ed.), John Wiley, 1991.

Chand, S., Schneeberger, H., "A Note on the Single-Machine Scheduling Problem with Minimum Weighted Completion Time and Minimum Allowable Tardiness", *Naval Research Logistics Quarterly* 33, 551-557, 1986.

Chand, S., Schneeberger, H., "Single Machine Scheduling to Minimize Weighted Earliness Subject to No Tardy Jobs", *European Journal of Operational Research* 34, 221-230, 1988.

Elleby, P., Fargher, H.E., Addis, T.R., "A Constraint-based Scheduling System for VLSI Wafer Fabrication", in *Knowledge-based Production Management Systems*, J. Browne(ed.), Elsevier Science Publishers, 1989.

Fisher, J., "Use of Nonfinancial Performance Measures", *Journal of Cost Management*, Spring 1992, 31-38.

Fox, M.S., Smith, S.F., "ISIS: A Knowledge-based System for Factory Scheduling", *Expert Systems* 1, 25-49, 1984.

Garey, M.R., Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP- Completeness*, W.H. Freeman, San Francisco, 1979.

Goldratt, E.M., *The Goal*, North River Press, 1986.

Graves, S.C., Meal, H.C., Stefek, D., Zeghmi, A.H., "Scheduling of Re-Entrant Flow Shops", *Journal of Operations Management* 3, 197-207, 1983.

Harrison, J.M., Holloway, C.A., Patell, J.M., "Measuring Delivery Performance: A Case Study from the Semiconductor Industry", in *Measures for Manufacturing Excellence*, R.S. Kaplan (ed.), Harvard Business School Press, 1990.

Dileepan, P., Sen, T., "Bicriterion Static Scheduling Research for a Single Machine", *OMEGA* 16, 53-59, 1988.

Kempf, K., Russell, B., Sidhu, S., Barrett, S., "Artificially Intelligent Schedulers in Manufacturing Practice", *AI Magazine* 11, No.5, 46-56, 1991.

Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B., "Sequencing and Scheduling: Algorithms and Complexity", in *Handbooks in Operations Research and Management Science Vol.4 Logistics of Production and Inventory*, S.C. Graves, A.H.G. Rinnooy Kan, P. Zipkin (eds.), North-Holland (1993).

Miyazaki, S., "One Machine Scheduling Problem with Dual Criteria", *Journal of the Operations Research Society of Japan* 24, 37-50, 1981.

Nelson, R.T., Sarin, R.K., Daniels, R.L., "Scheduling with Multiple Performance Measures: The One-Machine Case", *Management Science* 32, 464-479, 1986.

Potts, C.N., van Wassenhove, L.N., "An Algorithm for Single-Machine Sequencing with Deadlines to Minimize Total Weighted Completion Time", *European Journal of Operational Research* 12, 379-387, 1983.

Rodammer, F.A., White, K.P., "A Recent Survey of Production Scheduling", *IEEE Transactions on Systems, Man and Cybernetics* 18, 841-852.

Sen, T., Gupta, S.K., "A Branch and Bound Procedure to Solve a Bicriterion Scheduling Problem", *IIE Transactions* 15, 84-88, 1983.

Sen, T., Raiszadeh, F.M.E., Dileepan, P., "A Branch and Bound Approach to the Bicriterion Scheduling Problem involving Total Flow Time and Range of Lateness", *Management Science* 34, 254-260, 1988.

Smith, W.E., "Various Optimizers for Single-Stage Production", *Naval Research Logistics Quarterly* 3, 59-66, 1956.

Sze, S.M., *VLSI Technology*, McGraw-Hill, New York, 1988.

Van Wassenhove, L.N., Gelders, L.F., "Solving a Bicriterion Scheduling Problem", *European Journal of Operational Research* 4, 42-48, 1980.

Reactive Scheduling Systems

STEPHEN F. SMITH

Center for Integrated Manufacturing Decision Systems, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, sfs@cs.cmu.edu

Abstract

In most practical environments, scheduling is an ongoing reactive process where evolving and changing circumstances continually force reconsideration and revision of pre-established plans. Scheduling research has traditionally ignored this "process view" of the problem, focusing instead on optimization of performance under idealized assumptions of environmental stability and solution executability. In this paper, we present work aimed at the development of reactive scheduling systems, which approach scheduling as a problem of maintaining a prescriptive solution over time, and emphasize objectives (e.g., solution continuity, system responsiveness) which relate directly to effective development and use of schedules in dynamic environments. We describe OPIS, a scheduling system designed to incrementally revise schedules in response to changes to solution constraints. OPIS implements a constraint-directed approach to reactive scheduling. Constraint analysis is used to prioritize outstanding problems in the current schedule, identify important modification goals, and estimate the possibilities for efficient and non-disruptive schedule modification. This information, in turn, provides a basis for selecting among a set of alternative modification actions, which differ in conflict resolution and schedule improvement capabilities, computational requirements and expected disruptive effects.

key words: reactive decision-making, constraint-based problem solving, blackboard systems and production scheduling.

1. Introduction

The broad goal of manufacturing production management, like other resource-constrained, multi-agent planning problems, is to produce a coordinated behavior where demands are serviced in a timely and cost-effective manner. In most manufacturing environments, the construction of advance schedules is recognized as central to achievement of this goal; it enables anticipation of potential performance obstacles (e.g., resource contention) and provides opportunities to minimize their harmful effects on overall manufacturing system behavior. In practice, however, two factors confound the use of schedules as operational guidance. The first is the inability to extract useful guidance from generated schedules in any effective, systematic manner; most existing production planning and scheduling systems operate with models that ignore important operating constraints and conditions, and the correspondence of generated schedules to factory floor operations is missing. The second confounding factor is the dynamically changing nature of the production environment. Machines break down, materials fail to arrive on time, changing market conditions present unexpected production demands, etc., all of which work against attempts to follow prescriptive plans. In fact, the performance of the manufacturing organization ultimately hinges on an ability to rapidly adapt schedules to fit changing circumstances over time. In this respect, production planning/scheduling is not a static optimization problem, but an ongoing reactive process. Optimization objectives (assuring a good global manufacturing behavior) must continually be balanced with concerns of continuity in operations (because execution of a schedule sets a large number of interdependent processes in motion) and responsiveness (to keep the manufacturing system moving).

In this paper, we present an approach to incremental reactive management of schedules based on a view of scheduling as an iterative, constraint-directed process. Under this view, schedule revision/adaptation is driven by detection and analysis of conflicts and opportunities that are introduced by changes to current solution constraints. Constraint analysis is used to prioritize outstanding problems in the current schedule, identify important modification goals, and estimate the possibilities for efficient and non-disruptive schedule modification. This information, in turn, provides a basis for selecting among a set of alternative modification actions, which differ in conflict resolution and schedule improvement capabilities, computational requirements and expected disruptive effects. This approach to reactive scheduling is implemented in OPIS (OPportunistic Intelligent Scheduler), a knowledge-based system developed originally for manufacturing production scheduling. OPIS combines a modeling framework suitable for capturing essential operational constraints and objectives with a blackboard-based control architecture to provide a general infrastructure for configuring constraint-based schedule revision methods and strategies. The current OPIS manufacturing scheduler instantiates this infrastructure

with a specific set of methods and a constraint-based model for directing their application in responding to reactive scheduling problems. Experimental studies have demonstrated the viability and effectiveness of the scheduler in both generative and reactive scheduling contexts in a variety of complex manufacturing scheduling domains.

To provide a context for discussion, we first examine the characteristics and complexities of the reactive scheduling problem in practical domains. We then sketch the origins and evolution of the current OPIS scheduler, highlighting the basic concepts that underlie the approach and summarizing the results obtained. Next, we describe the principal components of the current OPIS scheduling architecture. This provides a basis for summarizing the heuristic methodology defined in the current OPIS manufacturing scheduler and illustrating its operation in responding to reactive scheduling problems. The approach is then summarized and contrasted with other recent work in reactive scheduling, and we conclude with a brief discussion of various ways in which the OPIS approach has been and is currently being extended and generalized for use in other application domains.

2. The Reactive Scheduling Problem

The general problem of interest in this paper is that of managing prescriptive solutions to scheduling problems. In brief, scheduling problems involve allocation of resources to the activities of multiple independent processes over time to achieve a targeted global behavior. Coordination of production in a factory, management of space missions, and transportation scheduling to support crisis management are representative examples. To be viable as operational guidance, a solution - or schedule - must first be *feasible*; i.e., it must satisfy the physical constraints in the domain relating to usage of resources and execution of processes. In practical domains, these constraints are often wide ranging and complex in nature. In manufacturing production environments, for example, resource allocation decisions must be consistent with capacity limitations, machine setup requirements, batching constraints on parallel use, work shift times, etc. Similarly, production activities have associated duration and precedence constraints, and may require the availability of multiple resources (e.g., machines, operators, tooling, raw materials).

However, feasibility alone is rarely the goal of scheduling; typically the task is one of optimizing (to the extent possible) a set of objectives and preferences. For example, processes to be coordinated typically have requested start and due dates, and one scheduling objective is to attend to these constraints. Because it may not be possible to satisfy all of these constraints (nor is it generally computationally feasible to determine precisely whether or not this is the case), the common operational objective is minimizing tardiness. Other global objectives (often conflicting) relate to the efficiency with which the processes are executed and resources are utilized:

minimizing wait time between constituent process activities, maximizing resource utilization, etc. In some cases, performance objectives can be approximated by tactical operating biases (or preferences). In a manufacturing context, for example, there might be a preference for a new machine over an older machine with overlapping capabilities because of reliability considerations. The *quality* of a schedule is a function of the extent to which it achieves (or effectively balances) stated objectives and preferences.

Scheduling research has traditionally focused on generating optimal solutions to classes of problems that make specific assumptions about the nature of domain constraints and objectives (e.g., [11]). Unfortunately, practical scheduling domains rarely meet these assumptions. But more generally, scheduling can rarely be treated as a static optimization problem. Aside from unpredictability in the execution environment, schedule generation in practice also tends to be a dynamic reactive process (particularly when multiple decision-makers are involved). An initial schedule is built, problematic or unsatisfactory aspects of the result are identified, requirements are relaxed or strengthened, schedule modifications are made and so on. Here, the current schedule provides the context for identifying and negotiating constraint changes (with the user or other scheduling agents). Although the focus is on improving the acceptability/quality of the solution, there is considerable pragmatic value placed on maintaining continuity in the schedules produced across iterations. Likewise, once execution begins, it is important to preserve continuity in domain activity while those changes are made that are necessary to ensure continued feasibility and attendance to overall performance objectives.

These pragmatic requirements argue strongly for approaches to scheduling based on incremental revision/adaptation of an existing schedule. It is this net change perspective that leads to what we refer to as the reactive scheduling problem. We assume that a schedule consists of a set of constraints that delineate (1) start time, end time and resource assignments for each activity of each process, and (2) available resource capacity over the scheduling horizon. The need for schedule revision arises in response to the introduction of new constraints or the removal of existing constraints, which might reflect the receipt of status/requirements updates from the environment or might be due to the results of prior modification actions. Schedule modification can be initiated for two purposes: (1) to restore the feasibility of a schedule now known to be infeasible because of the introduction of new conflicting constraints or (2) to attempt to produce a higher quality solution if one or more constraints respected by the current schedule have been relaxed. In the first case, modification is necessary to insure continued executability. In the second case, the potential gain in schedule quality may be weighed against its likelihood.

There are several characteristics of the schedule revision problem that influence the approach taken in OPIS toward its solution:

- It is generally not possible to bound the scope of change required to

the current schedule in advance. Given the tightly coupled nature of scheduling decisions, changes to one portion of the schedule often have ripple effects. Heuristic guidance can minimize this phenomenon, but problem combinatorics prevent its elimination. Schedule modification must necessarily proceed *opportunistically* (i.e., with the understanding that revision actions may have unforeseen interactions with other portions of the schedule that must subsequently be resolved).

- Striking an appropriate balance between attending to various scheduling objectives, minimizing disruption, and being computationally efficient when reasoning about possible modifications is a difficult task. There are often simple, fairly non-disruptive changes that can restore schedule feasibility. For example, if resource capacity is suddenly lost, affected activities could simply be delayed until a time in the schedule when required resource capacity is available (disregarding implications with respect to scheduling objectives such as minimizing tardiness). At the same time, however, it is not possible to enforce rigid bounds on schedule quality (e.g., degree of tardiness allowed) with assurance that a solution actually exists.

- What about exploiting the expertise of human schedulers? Although we do not discount the experience accumulated by veteran human schedulers, our experience in many manufacturing environments is that schedulers often fall prey to the complexity of interacting constraints and decisions, and tend to adopt myopic "fire-fighting" tactics (where extinguishing one fire ignites the next). Such tactics keep execution moving, but global system behavior deteriorates rapidly. In OPIS, use of knowledge about current problem structure (i.e., properties of current solution constraints) is advocated as an alternative basis for directing the schedule revision process.

3. Origins and Evolution of Approach

The approach to the reactive scheduling problem taken in the OPIS scheduler is rooted in earlier work with the ISIS job shop scheduling system[8, 9]. The ISIS scheduler was the first attempt to formulate and operationalize the view of scheduling as a heuristic, constraint-directed activity. ISIS emphasized complete representation of all constraints that impact operational decision-making, and a representational framework that recognized the conflicting and negotiable nature of many of these constraints. A representation of preference (i.e., relaxable) constraints was defined to encode knowledge relating to various factory objectives and operating preferences, including their relative importance, possible relaxations of preferred choices, the utility of each alternative, and the types of decisions that the constraint impacts. This knowledge about preferences was embedded in a larger relational framework for modeling the entities and physical con-

straints of the production environment. The OPIS modeling framework was built directly on these representational concepts.

ISIS also introduced a heuristic search framework for using constraints to guide the scheduling process. At the core of the ISIS approach was a beam search strategy for adding a new job (order) into the developing shop schedule (or alternatively revising the schedule of a job previously added to the schedule). Within this search, physical constraints (i.e., operation precedence constraints, resource requirements and availability) were used as a basis for generating alternative sets of decisions. Relevant preference constraints provided the basis for evaluation and pruning of alternatives at each step of the search, thus implementing a generative approach to constraint relaxation. This core search process was augmented in two ways to define the overall scheduling procedure. First, it was bracketed by rule-based analysis steps. Domain specific rules were applied both prior to any invocation of the search to fix specific relaxable constraints (e.g., specify "backward" scheduling to ensure satisfaction of due date) and after the search to assess results and propose constraint relaxations if appropriate (e.g., relax the due date and attempt "forward" scheduling if a feasible schedule cannot be found during backward scheduling). Second, this extended beam search procedure was embedded in an iterative, hierarchical control regime where additional types of constraints were considered at each successive level. After selecting the next job to schedule (or reschedule) on each iteration, a high level analysis of remaining resource availability was used to emphasize specific allocation intervals (through the introduction of additional preferences). After detailed forward or backward beam search scheduling, which incorporated these additional preferences, final local optimization to minimize WIP time was performed.

One difficulty encountered with this search architecture was its inflexibility with respect to strategic problem decomposition. Although the kernel heuristic beam search procedure provided a flexible basis for local search in the presence of diverse constraints and objectives, its placement within an overall decomposition framework based rigidly on relative job priority and stepwise construction of job schedules was found to impose significant limits on the system's ability to achieve a good compromise with respect to conflicting global objectives.[29]. The problem can be simply seen by considering two common factory scheduling objectives: minimizing work-in-process (WIP) time and maximizing resource utilization. The job-oriented problem decomposition framework of ISIS provides an opportunity to minimize WIP time, because subproblems are solved that contain all constraints involved in optimizing this objective. At the same time, a job-oriented decomposition works against the objective of optimizing resource usage because the constraints relevant to this tradeoff are spread across subproblems. A resource-oriented decomposition strategy, alternatively, provides the complementary opportunity to optimize resource usage (at the expense of leverage in minimizing WIP). The subproblems solved

in this case contain the competing requests of multiple jobs for a given resource, enabling resource setups to be minimized. Similar arguments against a fixed decomposition strategy can be made in reactive scheduling contexts. Localized revision of individual job schedules (the reactive strategy in ISIS) can provide a direct basis for resolving operation precedence violations (e.g., resulting from quality control failures and the subsequent introduction of "part repair" operations) with continued attendance to WIP minimization, but provides only indirect leverage, at best, in rearranging resource assignments to maximize throughput in response to an unexpected loss in resource capacity.

To broaden the range of constraints and objectives that could effectively be dealt with, the OPIS scheduler adopts a more flexible approach to problem structuring, referred to as *multi-perspective scheduling.* The concept of multi-perspective scheduling, as originally conceived, advocated the selective use of a complementary set of local scheduling methods, each conducting its search under different problem decomposition assumptions. An initial implementation (OPIS 0) focused on integrating use of the job-oriented beam search procedure of ISIS with a resource-oriented search procedure built around the "idle time" priority rule [19] and designed to maximize usage of substitutable resource groups. Schedules were generated according to a predefined strategy, first constructing a schedule for a pre-designated bottleneck resource group, and then completing the shop schedule on a job-by-job basis. Comparative experimental analysis carried out in the context of a specific Westinghouse job shop environment convincingly demonstrated the power of this configuration over both ISIS and a well-regarded resource-centered dispatch scheduling method in balancing weighted tardiness and WIP minimization goals over a broad range of factory conditions.[22].

Although OPIS 0 experiments confirmed the utility of incremental schedule construction from both resource-centered and job-centered decomposition perspectives, it did so under rigid and simplifying control assumptions. By presuming a static problem structure comprised of a single pre-identified bottleneck resource group, it was possible to perform the necessary computation to ensure feasibility of the partial schedule each step of the way (and avoid the complications of backtracking). However, dependence on any static assumptions about problem structure is ultimately confining (this was the original criticism of the ISIS search architecture). Resource bottlenecks are typically not stationary but float over time according to production characteristics (e.g., job mix, shop load); attendance to primary resource bottlenecks can lead to the emergence of secondary bottlenecks; and in some cases there is no dominant locale of resource contention in the overall manufacturing system. Each of these circumstances suggests different problem decomposition/structuring decisions, and indicate the need for dynamically determined schedule building strategies.

To dynamically control the use of local search methods oriented

around job and resource loci respectively, the OPIS 1 scheduler [27] introduced the concept of *constraint-based control.* The general idea here is that monitoring and analysis of characteristics of the evolving structure of solution constraints (in particular, flexibilities and inflexibilities in the solution space) can provide useful problem structuring knowledge. As foreshadowed in the design of the initial hard-wired, multi-perspective strategy of OPIS 0, resource contention was incorporated in OPIS 1 as the essential aspect of current solution structure upon which to focus the schedule building process. A resource capacity analysis procedure was defined and used in conjunction with the heuristic that decisions at "bottleneck" resources are the most critical to the overall quality of the solution and should be considered first[19, 15]. In moving to this *opportunistic* scheduling strategy, the OPIS 0 assumption of a guaranteed feasible partial solution at each intermediate solution state was also abandoned, giving rise to the possibility of inconsistent decisions. It was felt that repeated generation of "throw away" schedule extensions to ensure feasibility (as was done in OPIS 0) would not only become computationally over-bearing but would also add undesirable bias to the opportunistic scheduling strategy (because of the influence of the heuristics used to generate extensions). In OPIS, inconsistent intermediate solution states are seen as equivalent to inconsistencies arising from unexpected external events. They are reactive scheduling problems to be solved. In the OPIS 1 scheduler, a simple "schedule shifting" method was added to reconcile all detected conflicts and allow the schedule building process to proceed.

This opportunistic approach to schedule building was supported by an underlying system architecture based on standard principles of blackboard systems[7]. The OPIS 1 architecture reflected the central role of constraint management in the scheduling process and the constraint-based approach to coordinating local search. The design anticipated a wider array of solution constraint metrics upon which to base strategic control decisions and a larger repertoire of potential scheduling actions. Constraint analysis routines and scheduling methods were encapsulated as knowledge sources which, when triggered, operated on a globally accessible representation of the current solution. Strategic decision-making was localized within a single controller, that maintained and executed an explicit plan (i.e., an agenda of analysis and scheduling tasks) for solving the problem. The OPIS 1 architecture provided an initial infrastructure for investigating constraint-based, multi-perspective approaches to reactive schedule revision and adaptation.

Focus on the reactive scheduling problem necessitated a shift in perspective with respect to strategic control. Whereas problem solving in generative contexts can be driven top-down from global analysis of problem requirements and identification of critical decisions, the situation is different in reactive contexts. Here the starting point is a set of identified problems in the current solution (e.g., constraint conflicts), and focusing heuristics must be based on localized solution analysis. Moreover, control

decisions must balance pressing (re)optimization needs with potential opportunities to make revisions with limited non-local impact. Investigation of reactive scheduling strategies also sharpened understanding with respect to scheduling architecture requirements and highlighted problems in some of the assumptions made in the OPIS 1 scheduler. Most important was recognition of the inappropriateness of attempting to maintain a strategic control plan for solving a reactive scheduling problem. Because it is generally not possible to predict the non-local effects of a given conflict resolving action (e.g., the number and types of conflicts, if any, that will be introduced as a result of applying the action, the potential serendipitous effects of this action on other currently pending conflicts), it was rarely the case that an initially formulated control plan could be carried out to completion. Reassessment and revision of the agenda of pending tasks was typically required at each strategic step.

These considerations and experiences contributed to development of the OPIS 2 scheduler [25]. Work in reactive scheduling led to expansion of the set of scheduling methods, extending the search-based methods utilized in OPIS 1 to provide schedule revision capabilities and incorporating additional, more-specialized repair actions. The underlying scheduling architecture was revised to provide a more reactive, blackboard-based control framework. Within the OPIS 2 architecture, schedule generation and revision is uniformly cast as an iterative process of subproblem formulation and subproblem solution, opportunistically directed by analysis of current problem structure. A specific heuristic model, which maps the optimization needs and opportunities implied by specific reactive scheduling states to the differential capabilities of the expanded set of revision methods, was developed and implemented as a means of validating the approach [20]. A series of experiments carried out in the context of an IBM computer board assembly and test line demonstrated the comparative advantage of this reactive model over several less flexible revision strategies as well as a random selection model biased by the choice percentages observed using the model across all experiments (to verify that the model did, in fact, encode useful knowledge) [25]. Performance was evaluated in this study with respect to weighted criteria reflecting optimization, stability, and efficiency objectives across a diverse range of reactive problems involving unexpected resource loss and job delays due to quality control failures. More recent experimental analysis has evaluated this reactive model in support of generative decision-making[1]; reruns of the original multi-perspective job shop experiments with the OPIS 2 scheduler, using dynamic, bottleneck-based problem decomposition and the reactive model to resolve conflicts introduced along the way, yielded significant further improvements in schedule quality over the original hard-wired multi-perspective strategy.[1]

In the next few sections, we summarize the organization and opera-

[1]The reader is referred to [22, 20, 25, 1] for details of the experimental results obtained with variants of the OPIS scheduler.

tion of the OPIS 2 scheduling system (referred to hereafter simply as OPIS). We first describe the infrastructure for constraint-based, reactive scheduling provided by the OPIS scheduling architecture . We then turn attention to the methods and heuristics implemented within this architecture in the current OPIS reactive scheduler.

4. The OPIS Scheduling Architecture

Figure 1. schematically depicts the blackboard-based control architecture defined in OPIS. The architecture presumes a collection of base scheduling methods (or knowledge sources) that carry out designated *scheduling tasks* and make changes to a commonly accessible representation of the current solution. An additional "model update" knowledge source is invoked upon receipt of external notification of constraint changes (e.g., new requirements, execution status updates) to reflect their consequences on the current solution. The introduction of changes to the current schedule (whether they are externally imposed or due to execution of a scheduling task) results in the posting of *control events* in a global description of the system's current control state. The control state at any point characterizes the set of outstanding problems that remain (i.e., current conflicts in the schedule, set of commitments that remain to be made, unexplored opportunities for improving the solution). A separate set of analysis knowledge sources extends this control description to provide the information necessary to support the formulation of subsequent scheduling tasks.

Two additional system components provide the distinguishing characteristics of the OPIS architecture and the infrastructure for opportunistic, multi-perspective scheduling: a *schedule maintenance* subsystem, which incrementally maintains a representation of current solution constraints, and *a top level manager (TLM)*, which holds responsibility for coordinating the use of scheduling, analysis and model update methods, and implements an event-driven control cycle. The former provides both a basis for analyzing aspects of the current scheduling state and a means for communicating scheduling constraints among different formulated subproblems. The latter defines a structure for specifying and a mechanism for applying the control knowledge necessary to implement constraint-based scheduling strategies.

Both the representation of the schedule maintained within OPIS and the methods incorporated to analyze and modify this schedule are defined relative to an underlying domain model, which contains a specification of the constraints and objectives on process execution and resource allocation that must be accounted for in the target environment. Before considering the schedule management subsystem and the TLM in more detail, we first summarize the structure of domain models constructed within the OPIS modeling framework.

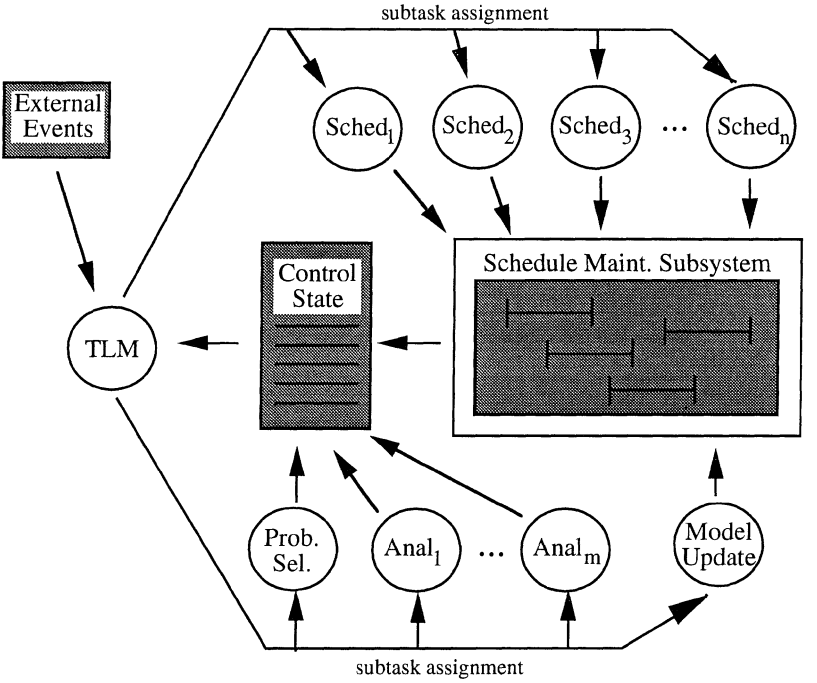


Figure 1. The OPIS Scheduling Architecture

4.1. Modeling Framework

Within a OPIS domain model, prototype process descriptions (e.g., manufacturing production plans for various part types) are represented as hierarchies of operations, with aggregate operations designating either more detailed sub-processes (i.e., sequences of operations) or sets of exclusive alternatives. Operation descriptions at any level specify precedence relations (predecessor and successor operations), duration constraints, and resource capacity and setup requirements. Resources are correspondingly represented hierarchically, with atomic resources grouped into increasingly larger resource pools, to provide a description of resource allocation constraints (e.g., available capacity, hours of operation) at each level of abstraction in the prototype plans. Thus, for example, if a disjunctive leaf operation specifies usage of a particular manufacturing machine as a resource requirement, the aggregate operation to which it is related at the next level specifies usage of capacity in an encompassing machine group as a resource requirement. This hierarchical domain model provides a structure

for representing and maintaining solution constraints at different levels of precision; in reactive contexts this hierarchical model provides one basis for reasoning about the scope of potential schedule modifications.

The OPIS modeling framework provides an extensible set of primitives for specifying a wide range of constraints on resource allocation. Resource representations enable specification of unit capacity resources, which must be allocated exclusively to a given process (e.g., a machine), batch capacity resources, which can simultaneously be allocated to multiple processes over the same interval (e.g., an oven), and a variety of disjunctive and conjunctive aggregate capacity resources, where capacity can simultaneously be allocated to multiple processes without temporal synchronization (e.g., machine, operator groups). Resource setup constraints can be modeled as temporal delays, expressed as arbitrary functions of the operations that consecutively utilize a given resource. Operation durations, similarly, can be expressed as functions of work content (e.g., the number of parts in the batch being processed). Work shift constraints can additionally be imposed on resource availability, with allowance for preemption of process execution across non-working hours. Priorities and priority classes can be associated with process requests (e.g., orders for manufactured parts) in addition to requested release and due dates. The utility-based representation of preferences originally developed in ISIS provides a basis for specifying idiosyncratic biases with respect to choice sets defined in the domain model (e.g., preferences over substitutable resource groups). Details of the OPIS representational framework can be found in [26].

4.2. Schedule Maintenance

To provide a representation of the current schedule, an appropriate prototype plan in the domain model is instantiated for each job or process to be accomplished in the current scheduling horizon. These instantiated plans designate the set of process operations that must be scheduled to obtain a complete solution (actually a superset, because some instantiated operations designate alternatives that will become undefined as choices are made). Given these instantiated plans and the resource hierarchy defined in the domain model, the schedule maintenance subsystem incrementally maintains the following solution constraints:

- the current time bounds (an earliest start time, latest end time pair) on the execution of each instantiated operation that has been or might be scheduled, and
- a specification of the current available capacity of resources at each level of the hierarchy over time at each level in the hierarchy (represented as an ordered sequence of intervals of the form *(st, et, capacity available)* that covers the scheduling horizon).

As additional scheduling decisions are made or the constraints implied by external status updates are introduced, the schedule maintenance system

combines these new constraints with the constraints on process execution and resource utilization defined in the underlying domain model and specified problem constraints (e.g., job release and due dates). This results in an updating of the time bounds and available capacity representations, respectively, of related operations and resources at all defined levels of abstraction. Thus, an unscheduled operation's time bounds at any point reflect the set of allocation decisions compatible with domain and problem constraints, and any scheduling decisions that have been made.

Constraint propagation in response to schedule changes can lead to the detection of two types of conflicts:

- time conflicts - situations where either the time bounds or scheduled execution times of two operations belonging to the same process instantiation violate a defined temporal precedence constraint, or the time bounds of a single operation violate a rigid absolute bound on process execution.[2]

- capacity conflicts - situations where the resource requirements of a set of currently scheduled operations exceed the available capacity of a specific resource over some interval of time.

The recognition of conflicts signals the need for schedule revision. Detected conflicts are posted in the current control state as *elementary conflict events* which require subsequent scheduling attention.

Constraint propagation can also lead to detection of rescheduling *opportunities*, situations where time and capacity constraints are loosened by introduced schedule changes. In the current implementation, such situations are treated in a somewhat specialized manner; *opportunity events* are posted only in response to changes originating from external events that imply additional resource capacity (e.g., cancellation of a process request) to ensure that a rescheduling process is triggered. The final type of control event that can be posted by the schedule management subsystem is an *incomplete hypothesis event*, signifying that some set of scheduling decisions still remains to be made. Details of this approach to schedule maintenance can be found in [14].

4.3. Strategic Control

Events posted by the schedule management subsystem are responded to within a control cycle defined by the TLM. The TLM control cycle identifies four stages of control decision-making that must occur in specifying the next scheduling action to perform and correspondingly four types of knowledge sources necessary to support the process:

[2] Some types of imposed time constraints, in particular due date constraints, are specified as relaxable instead of rigid. Violations of such relaxable constraints are not seen as conflicts, but are instead interpreted as relaxation decisions made by the originator of the change. In such cases, time bounds of other temporally related operations belonging to the same process are updated to reflect this newly determined end time bound.

- *event aggregation* - The first decision-making step contributes to selection of the particular control event (or events) of those currently posted to serve as the focal point of reaction on the current cycle. It is often the case that individual events are related in some manner and would be better addressed simultaneously. During event aggregation, knowledge of such relationships is applied to the set of posted events. In cases where specific relationships are detected, *aggregate* events are created and added to the list of posted events in the current control state.

- *event prioritization* - After this preprocessing of posted control events, prioritization heuristics are applied to select the specific event to be responded to in the current cycle. All events other than the highest priority event are left pending until the next cycle.

- *event analysis* - Having identified a focal point for problem solving, the next step in the control cycle is problem analysis. The goal of this step is to summarize essential aspects of the current solution state (e.g., the relative looseness or tightness of current time and capacity constraints), providing a basis for determining how to best respond to the event. Analysis results are appended to the event description in the current control state.

- *subproblem formulation* - During the last step, subproblem formulation knowledge is applied to the results of problem analysis, resulting in generation of a particular *scheduling task* to execute. A scheduling task designates (1) a particular component of the overall schedule to extend or revise, (2) a particular scheduling method to apply, and (3) depending on the KS selected, appropriate parameterization of the solution procedure.[3]

Once an appropriate scheduling task has been determined, it is carried out, the results are introduced into the current solution, and the TLM control cycle repeats. When, on any given cycle, the set of posted events becomes empty, a complete and consistent solution has been obtained and the process terminates.

5. Constraint-Based Schedule Repair in OPIS

The OPIS scheduling architecture allows specification of a range of methods and heuristics for addressing the reactive scheduling problem. The current OPIS manufacturing scheduler implements one such configuration of methods and heuristics, emphasizing principles of constraint-based focus

[3]It is possible in some circumstances for the subproblem formulation step to produce a sequence of scheduling tasks. In this case, all specified tasks will be executed before further consideration of the current control state.

of attention and multi-perspective scheduling discussed earlier in the paper. In this section, we examine the various heuristic components of this constraint-based repair methodology. We start with an overview of the base methods defined as strategic schedule revision alternatives.

5.1. Strategic Alternatives

The set of possible modification actions available in the current OPIS scheduler range from general heuristic search procedures oriented toward generating and revising sets decisions associated with a specific process or resource to more specialized revision procedures for sliding schedule components forward or backward in time and performing pairwise resource assignment exchanges. In describing these methods below, we restrict attention to describing revision capabilities, and characterizing their differential behavioral characteristics.

The **Order Scheduler (OSC)** provides a method for revising the schedule of some contiguous sequence of operations in the plan of a given process (e.g., the plan associated with a given manufacturing order). Revision of a designated process (or subprocess) schedule is accomplished by first retracting the current time and resource assignments of each constituent operation (i.e., releasing previously allocated intervals of resource capacity), and then applying the augmented beam search strategy of ISIS (see Section 3) to determine new resource assignments and execution intervals for the operation sequence. In addition to identifying the specific subprocess to be revised, an OSC scheduling task also designates a level of "visibility" with respect to resource availability. The search can be constrained to consider only execution intervals for which resource capacity currently exists, which we designate as the complete visibility (CV) OSC, or can be allowed to consider capacity allocated to lower priority jobs as available, which we designate as the prioritized visibility (PV) OSC. These two modes of operation are illustrated in Figure 2. Because prioritized visibility search admits the possibility of introducing additional capacity conflicts into the schedule (leading to "bumping" of lower priority processes), a decision to invoke the PV-OSC trades off potential additional disruption for some ability to perform resource-based optimization.

The **Resource Scheduler (RSC)**, provides a second general revision method, in this case for resequencing operations on a designated resource (or substitutable resource group) from a specified point in time onward so as to consistently accommodate a designated set of conflicting operations. Schedule revision is accomplished by "assuming" that the schedule must be completely regenerated from the revision start time onward and applying an iterative forward-dispatch search procedure to accomplish this goal. However, the decisions in the current schedule are not actually retracted prior to invoking this procedure; rather operations in the current schedule are only retracted when they are chosen to be "dispatched on a resource" on a given dispatch cycle. After all designated conflict op-

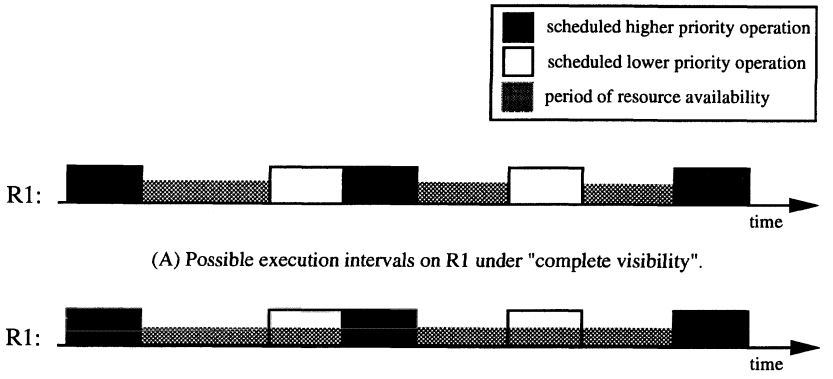


Figure 2. OSC visibility of allocation decisions

erations have been rescheduled, the procedure terminates on the first cycle where retraction and re-insertion of the chosen operations leave the overall resource schedule consistent. Thus, RSC attempts to preserve as much of the original resource schedule as possible while it resolves the problem at hand. The dispatch cycle itself makes selective use of a collection of priority rules to balance weighted tardiness and setup minimization concerns when selecting among alternative resource assignment and operation dispatch decisions. A detailed description of these heuristics can be found in [22]. The RSC method is designed from the assumption that contention for the resource(s) to be rescheduled is high; use of a dispatch-based search framework presumes that it is not necessary to consider insertion of resource idle time, and places emphasis instead on efficient resource utilization. Since revision of the schedule of a resource (or resource group) by RSC typically results in some amount of resequencing of scheduled operations (forcing some to be scheduled later than before) it is possible that its application will introduce new time conflicts with downstream process operations into the overall schedule.

The **Right Shifter (RSH)** implements a considerably less sophisticated reactive method that resolves conflicts by simply "pushing" the scheduled execution times of designated operations forward in time ("jumping" over any scheduled operation on the same resource whose end time falls before the end of the shift). Execution of these designated shifts can introduce both time conflicts (with downstream operations belonging to the same process) and capacity conflicts (with operations scheduled downstream on the same resource). However, these conflicts are internally resolved by recur-

sively propagating the shifts through resource and process schedules to the extent necessary. Thus, the RSH will not introduce any new conflicts into the overall schedule. Figure 3 shows the result of an RSH action to resolve a time conflict involving $op-a1$ by shifting its scheduled start time on $R1$. In this example, each process i follows the operation sequence $op-i1 \rightarrow op-i2$ on resources $R1$ and $R2$, respectively.

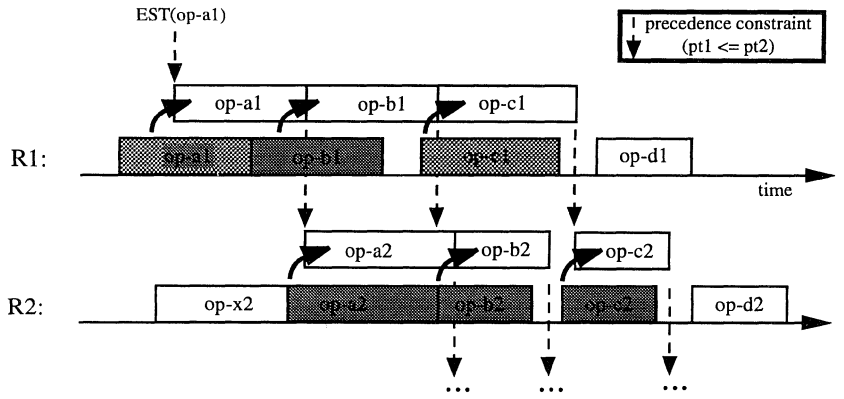


Figure 3. Application of RSH to $op-a1$ on $R1$

The **Left Shifter (LSH)** provides a similar but totally non-disruptive reactive method that "pulls" operations backwards in time (i.e., closer to execution) to the extent that current resource availability and temporal process constraints will permit. The method proceeds by sliding operations on a designated resource R to exploit an identified interval of available resource capacity (and any capacity intervals created by this sliding), and then recursively applying the procedure to the resources associated with the successor operations of processes who have had their scheduled execution interval on R changed. The recursion terminates whenever a downstream resource schedule is encountered that does not provide opportunities for left shifting or when process schedules have been completely traversed. Within the current implementation LSH is used exclusively for responding to opportunity events. Figure 4 shows the result of a LSH action invoked on resource $R1$ upon indication that $R1$ has become available earlier than expected. In this case, $op-a1$ is first rescheduled to start as soon as $R1$ is available, $op-c1$ is then shifted into the time interval vacated by $op-a1$ (because $op-b1's$ earliest start time constraint does not allow it to be moved), and finally $op-d1$ is shifted left as far as possible.

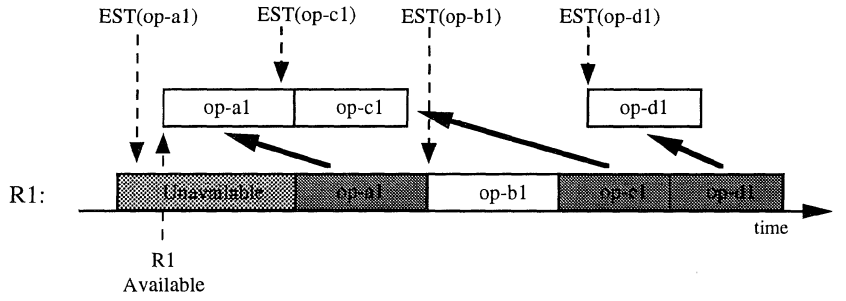


Figure 4. LSH revision in response to unexpected resource availability

The **Demand Swapper (DSW)** implements a final revision method based on pairwise exchange of the respective resource assignments and execution intervals of two similar processes. It is applicable in situations where a given process has unexpectedly been delayed and is now expected to be tardy. When invoked, DSW first identifies a set of suitable candidate processes for exchange (i.e., similar processes that are currently scheduled to complete ahead of their due dates). If at least one candidate is found, then an exchange of the remaining unexecuted portion of the problematic process's schedule with the corresponding portion of the schedule of another is made to minimize their combined tardiness. In essence, this action has the effect of redirecting the pair of processes to fulfill each other's respective demands. Note that the DSW is not necessarily a conflict resolution strategy. It is more appropriately viewed as a scheduling action that improves the character of the conflict. Figure 5 illustrates the result of applying a DSW action, in this case in response to the insertion of an extra "rework" operation into $Process1's$ operation sequence and the subsequent detection of a precedence violation between operations $rework - 1$ and $op - 1 - 2$. The depicted exchange of downstream execution intervals leaves the schedule of $Process1$ conflict free and ending with a small amount of tardiness. The precedence violation has moved to $Process2's$ schedule, but given that $Process2$ was originally scheduled to finish well ahead of its due date, there is now additional slack available for resolution of the conflict.

5.2. Responding to Conflicts

The above revision methods provide a range of capabilities for responding to reactive scheduling problems, and the strategic control problem faced by the scheduler is that of intelligently mapping relevant capabilities to detected problems. As indicated in Section 4, three general sources of knowledge are required by the TLM to solve this control problem: the first supporting selection of the particular conflict or set of conflicts to focus on

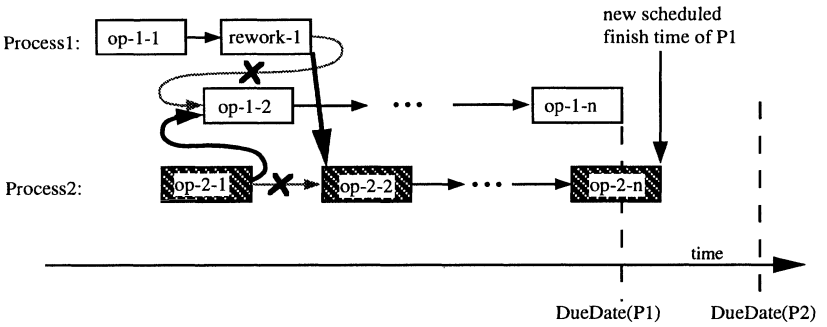


Figure 5. DSW revision in response to an unexpected process delay

next, the second providing characteristics of essential aspects of the current solution state, and the third concerning formulation of the most appropriate revision action to take to resolve this conflict (or set of conflicts).

5.2.1. Conflict Selection

Posted conflict descriptions provide basic characterizations of problems indicating the type of conflict (time or capacity), the operation(s) whose current commitments are in conflict, the resource (or resources) involved, the processes (or jobs) involved, the start time of the conflict and its temporal magnitude. Given this information, a variety of criteria can be defined for aggregating and prioritizing the current set of posted conflicts on any given revision cycle. The heuristics incorporated in the current scheduler reflect our experiences to date.

With respect to aggregation of posted elementary conflicts for simultaneous consideration, the heuristics currently utilized emphasize recognition of two types of relationships in posted elementary capacity conflicts that have been observed to occur with some regularity in specific reactive contexts.[4] The first relationship that triggers aggregation is based on *commonality in the resources involved* in elementary capacity conflicts. It is often the case that two or more capacity conflicts involving the same resource have intersecting sets of conflicting operations. For example, in the simple case of a unit capacity resource, if a given operation is scheduled over an interval that spans the scheduled execution times of several other operations previously allocated to that resource, individual capacity conflicts will be detected for each pair of operations. This situation is illustrated in Figure 6, which depicts (in Gantt chart form) portions of the schedules of

[4] We have experimented with various relationships for aggregating time conflicts but have as yet found none that add substantial value to rescheduling performance.

two unit capacity resources $R1$ and $R2$, and indicates two capacity conflicts CC_1 and CC_2 in $R1's$ schedule that involve $op - x1$. Because problems of over-allocation often require resource resequencing, it makes little sense to consider these conflicts individually.

The second situation under which elementary capacity conflicts are aggregated is based on *commonality in the processes involved* in the conflicts. This aggregation is motivated by a type of situation that can result from execution of a PV-OSC revision task (the only modification action that can introduce additional capacity conflicts). If PV-OSC determines new scheduling decisions for a given process that bump a lower priority process with a similarly structured plan (as is common in manufacturing environments with flow shop characteristics), then it is quite possible for capacity conflicts to be introduced on several resources which involve operations belonging to these same two processes. This situation is also depicted in Figure 6, where it is assumed that jobs are processed first on $R1$, then on $R2$. Conflicts CC_1 and CC_3 involve the same two jobs at two consecutive process steps. Because multiple components of the same process schedule now require revision, it is natural to consider the aggregate problem.

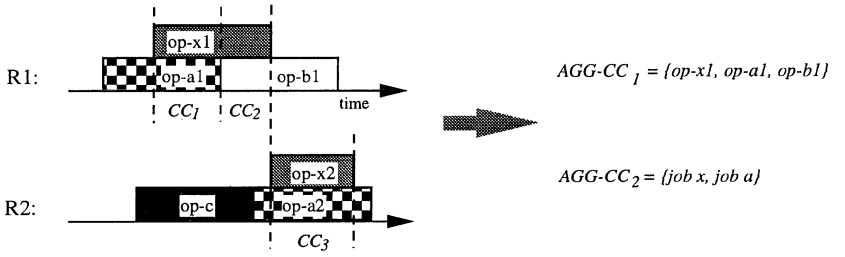


Figure 6. Aggregation of conflicts

Recognition of either relationship leads to the generation of aggregate events (as depicted in Figure 6) and their introduction into the list of currently posted events. Note that distinct aggregate events are not necessarily mutually exclusive groupings of elementary events. Any created aggregate event that is not selected as the current focal point for reaction is discarded.

Within the current scheduler, events are selected (prioritized) first as a function of event type. Aggregate conflicts are higher priority than elementary conflicts, and process-based aggregations (by their linkage to an ongoing reactive strategy) are higher priority than resource-based aggregations. Elementary time and capacity conflicts are of equal priority.

Opportunity events are lowest priority and are only selected in conflict free situations (because conflict resolution can actually exploit any posted opportunities). The second selection criteria applied (in situations where event type does not identify a unique event) is the urgency of the event (i.e., temporal proximity to execution).

5.2.2. Conflict Analysis

Determination of the most appropriate revision action to undertake to resolve a given conflict requires knowledge about the continued validity of various scheduling decisions, pressing (re)optimization needs, and current sources of rescheduling flexibility. The base assumption underlying the OPIS scheduler is that analysis of current problem structure can provide this knowledge. In this section, we describe the set of metrics that are currently computed for this purpose. These metrics summarize various characteristics of current solution constraints, and computations are confined to a local region of the solution containing the focal point conflict. A *conflict horizon*, defined as an interval that temporally spans the conflict by a pre-specified margin, restricts attention to a subset of operations in addition to those directly involved the conflict.[5]

Two metrics are defined to estimate the severity of conflict itself, and five others are defined to characterize the tightness or looseness of current time and capacity constraints in the schedule. Several are defined with respect to a particular resource involved in the conflict, designated R_C. For all types of capacity conflicts except process-based aggregate conflicts, R_C is the single resource that is over-allocated (which might well be an aggregate capacity resource representing a pool of more atomic resources). Unless otherwise stated, we will assume that relevant metrics are computed relative to each constituent $< conflict, R_C >$ pair in the case of a process-based aggregate conflict. In the case of a time conflict, which designates a precedence violation between two consecutive process operations, we assume that R_C is the resource assigned to the downstream operation. We also designate $Confl_C$ as the set of operations whose commitments are in conflict (the union over all constituent conflicts in the case of all aggregate conflicts). Figure 7 graphically depicts a portion of a schedule for aggregate resource R_C that contains a capacity conflict, identifying both the conflict horizon and $Confl_C$.

Conflict Duration is defined as the temporal magnitude of the inconsistency (e.g., $t2-t1$ in Figure 7) normalized with respect to the average duration of all operations scheduled on R_C within the conflict horizon (e.g., the average duration of operations $op-b$ through $op-h$ in Figure 7). This metric provides one indicator of the continuing validity of R_C's schedule. Appealing to sensitivity analysis[2, 17], if the duration is low, then it is reasonable to assume that the sequencing decisions previously made at R_C

[5]This use of metrics is quite similar to the concept of "textures" described in [10], although the focus there is in generating schedules.

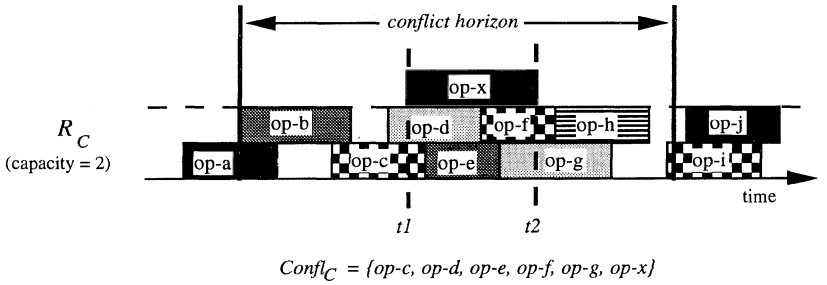


Figure 7. A capacity conflict

remain valid. The problem lies only in the timing details. If the duration is high, however, this assumption is not valid.

Conflict Size is defined as $\mid Retract_C \mid$, where $Retract_C$ is an identified (minimal) set of operations in $Confl_C$ that must be retracted (and hence rescheduled) to restore consistency (in the case of process-based aggregate conflicts, $Retract_C$ is computed for each constituent capacity conflict and conflict size is the average size of these sets). $Retract_C$ is constructed with the assumption that the operation whose commitment created the conflict will not be moved, and this set is retained for later use in formulating revision tasks. In Figure 7, for example, $Retract_C = \{op-d, op-f\}$ and $\mid Retract_C \mid = 2$ if we assume all operations are of equal priority. If the conflict size is low, then it is reasonable to assume that most sequencing decisions relative to R_C are valid. For example, if conflict size is 1, then the operation in $Retract_C$ may be the only one out of sequence. If the number is high, then sequence optimization at R_C is an important concern.

Resource Idle Time is defined as the average amount of capacity available at R_C over the conflict horizon (recall we are typically speaking of an aggregate resource). Figure 8 indicates periods of available capacity (or resource idle time) in the case of the conflict contained in Figure 7. If average idle time is low, indicating a bottleneck situation, then optimization of R_C's schedule to achieve maximum throughput is a primary concern. If average idle time is high, then resource-based optimization is unimportant. Resource Idle time is also computed for any disjunctive aggregate capacity resource that contains R_C in the underlying hierarchical model. An increase in resource idle time moving upward in the resource hierarchy indicates flexibility to assign alternative resources to operations in $Confl_C$.

Local Upstream Slack measures the flexibility in the scheduled start times of the operations scheduled on R_C. The upstream slack of a given job is defined simply as the difference between the scheduled start

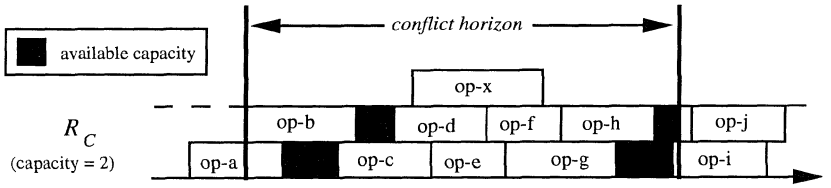


Figure 8. Periods of idle time within the conflict horizon

time of its operation on R_C and the scheduled (or actual) end time of its immediately preceding operation, and local upstream slack is the average slack over all jobs scheduled on R_C within the conflict horizon. Figure 9 expands the schedule given in Figure 7 to include portions of the upstream and downstream schedule (on resources R_i and R_j respectively), and indicates the upstream slack associated with jobs c, e, g, and h (in this figure the operation sequence $op - i' \rightarrow op - i \rightarrow op - i''$ is assumed for each job i). If the local upstream slack of operations requiring R_C is high, then there are opportunities for resequencing on R_C. It might be possible to place operations into the "holes" of available capacity that will be vacated by operations in $Confl_C$.

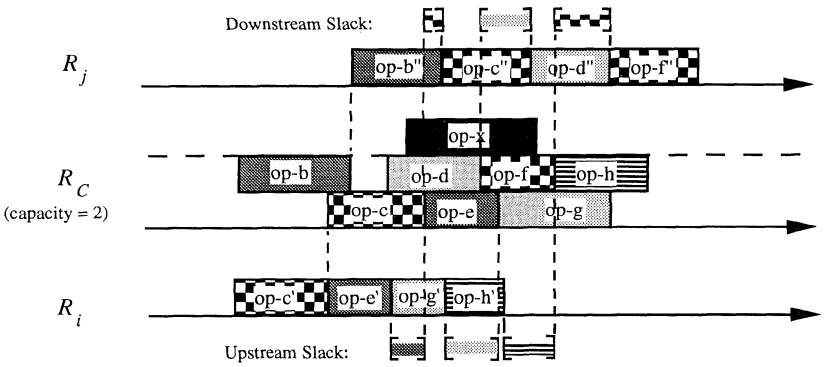


Figure 9. Upstream and downstream slack

Local Downstream Slack, alternatively, captures the local flexibility in the scheduled end times of operations currently scheduled on R_C.

In defining this measure, we appeal to an assumption concerning characteristics of a good schedule, namely that good schedules will exhibit queue times only before bottleneck resources. Given this assumption, we define local downstream slack to be the average of the durations of the first scheduled delay in the downstream schedules of each process/job scheduled on R_C within the conflict horizon. If there are no scheduled delays in the schedule of a given process, then there is no local slack. Figure 9 (top) indicates the downstream slack for jobs b, c, d, and f scheduled on R_C in the simple case where there is only one successor operation. If downstream slack is low, then downstream resource contention is not likely to be severe (i.e., there are no apparent downstream bottlenecks). If downstream slack is high, there is evidence of at least one downstream bottleneck and optimization of resource schedules further downstream can be important.

Projected Lateness defines another average measure of temporal flexibility, this time relative to the operation(s) in $Confl_C$. The projected lateness of an operation is defined similarly to local downstream slack, except now we are interested in the difference between the earliest time that the process/job can arrive at the downstream bottleneck and its scheduled start time on that resource. If there is no downstream bottleneck, then we are interested in the difference between the earliest the process can finish and its due date. If the projected lateness of an operation in $Confl_C$ is negative (i.e., the operation is still "early" relative to its current deadline), then resource-based optimization is unimportant. If the lateness is positive, then resource-based optimization is important.

Variance in Projected Lateness is defined with respect to all operations scheduled on R_C within the conflict horizon. This provides an indication of the opportunities for pair-wise optimization of process/job schedules. If the variance is high, then it might be possible to trade off positive and negative projected latenesses of specific processes by swapping demands.

5.2.3. Subtask Formulation

Determination of how to best resolve a given conflict is based on a qualitative model that combines the above stated implications of computed analysis metrics with knowledge of the differential optimization and conflict resolution capabilities of alternative revision actions. Given the results of conflict analysis, the model yields the revision method whose behavioral characteristics best match recognized revision needs and opportunities. The reasoning process underlying construction of the model is illustrated in Figure 10. In this case, the presence of a "large" conflict duration indicates the need for some amount of resequencing at R_C. Because resource idle time at R_C is "low", efficient utilization of R_C to maximize throughput is needed. RSC is the strongest candidate from the standpoint of these needs, with the potential disadvantage of introducing time conflicts with downstream process operations. However, because upstream slack is "high", there are

opportunities for non-disruptive sequence changes at R_C. Hence, RSC is selected as the method to apply.[6]

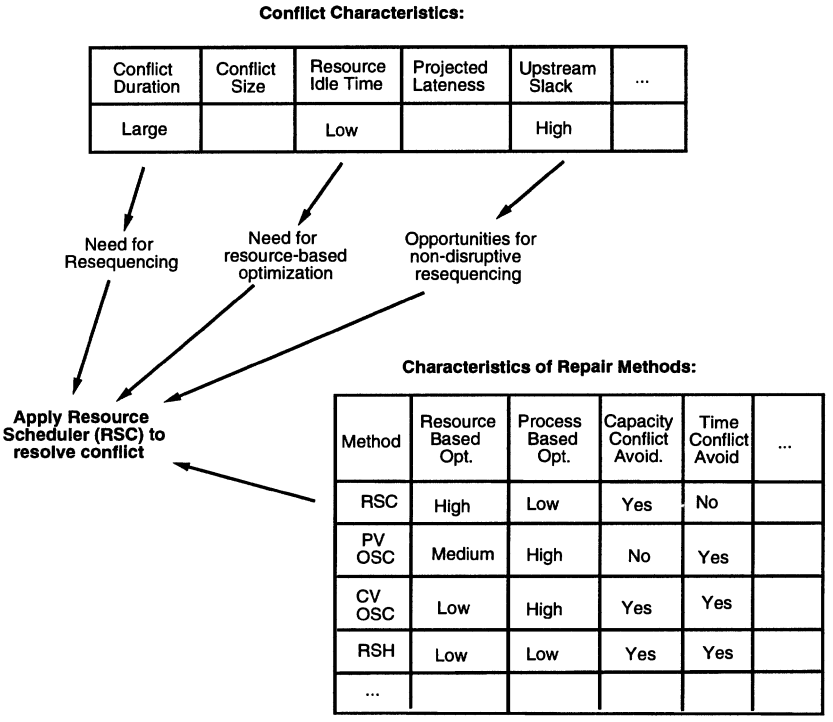


Figure 10. Mapping Analysis Results to Repair Actions

The set of rules that make up the constructed model is given in Table 1. Only one rule does not specify a unique choice. In this case, DSW is always selected first (because it typically acts to improve the character of the conflict), and then removed from consideration on the next cycle. As indicated previously, handling of opportunity events is restricted to those reflective of unexpected gains in available resource capacity. Such events are responded to only after a feasible schedule has been obtained, and always by applying LSH.

[6]Currently, simple thresholds on conflict analysis metric values are used to produce qualitative interpretations.

Confl. Dur.	*Confl. Size*	*Idle Time*	*Upstr. Slack*	*Proj. Late.*	σ^2*(P.L.)*	*Action*
small						RSH
large		high		−		CV-OSC
large		high		+	high	DSW, PV-OSC
large		high		+	low	PV-OSC
large	large	low				RSC
large	small	low	high			RSC
large	small	low	low			PV-OSC

Table 1
Action Selection Model

Formulation of a scheduling task involving the selected action requires determination of scope (i.e., what particular set of scheduling decisions is to be revised). This decision is also based on information provided by conflict analysis, as well as knowledge relating to prior modification actions. Because revision methods differ in the types of schedule components they manipulate, decisions about scope depend on the specific method selected. If either OSC and RSH actions are to be applied to resolve a capacity conflict, then the process (or processes) to be rescheduled must be selected. Here the operation whose commitments originally introduced the conflict are left intact; scheduling tasks are created for the processes associated with each operation contained in the set $Retract_C$ (computed during conflict analysis).[7] For OSC actions, it is also necessary to delineate which portion of its downstream schedule to revise. If downstream slack is high (indicating the presence of downstream bottleneck resources), then scope is limited to the portion of the process's plan that precedes the downstream bottleneck operation.

In the case of RSC actions, decisions about scope concern the level of aggregation of the focal point resource (or, equivalently, how large a set of substitutable resource schedules to revise). This decision is a function of the increased rescheduling flexibility that can be gained by broadening scope. If a disjunctive aggregate resource is found moving upward the hierarchical model with significantly higher estimated resource idle time than that of R_C, then that resource is selected as the focal point of the RSC action; otherwise scope is restricted to R_C. DSW tasks, finally, require designation of a set of processes to consider, which is taken to be the set of operations scheduled on R_C over the conflict horizon.

5.3. An Example

To illustrate the behavior of the OPIS reactive scheduler, we consider a sample reactive scenario involving response to an unexpected loss in

[7]Given the current action selection model, $| Retract_C |$ is rarely > 1 when OSC or RSH is selected.

resource capacity. Figure 11 graphically depicts (again in Gantt chart form) the current schedule for a portion of a hypothetical factory. The diagram indicates machine and time assignments of operations for nine jobs (or orders) covering three stages of the manufacturing process. The operations associated with each particular job are related by temporal precedence constraints which are designated by directed arcs in the diagram. The pattern used to designate the operations of a job reflects the job's "part type", which has implications with respect to machine setup requirements at different stages of the manufacturing process (there are four types of parts being manufactured in this example). We assume in this example that all assigned machines are unit-capacity resources (i.e., capable of servicing only one job at a time); thus, the time line associated with a given machine in the diagram indicates either a scheduled operation, a scheduled setup operation (represented as a black rectangle), or a period of availability (represented as blank space).

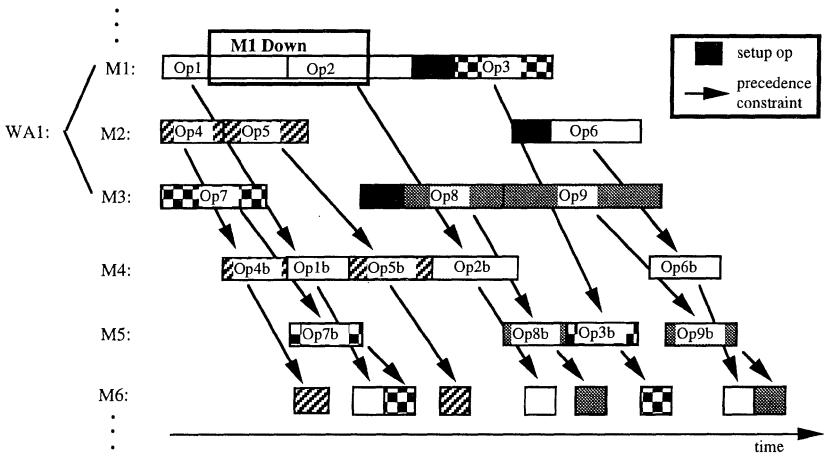


Figure 11. Unexpected Machine Breakdown

The three stages of the manufacturing process covered in the diagram impose the following allocation constraints. Machines *M1, M2,* and *M3* are organized into work area *WA1* and are utilized in the first stage of the process. Each of these machines is capable of manufacturing any of the four part types; however, a machine setup must be incurred in switching from one part type to another. Machines *M4* and *M5* are used in the second stage of the process. Each of these machines is specialized to the

manufacture of two unique part types and there is no setup required for part changeovers. Machine *M6*, finally, is used in the third stage for all part types (again with no setup requirements).

As indicated in Figure 11, machine *M1* is expected to be down for an interval that overlaps the ongoing execution of *Op1* and the scheduled execution of *Op2*. In this case, the already completed portion of *Op1* is salvageable. Introduction of these newly known constraints into the current solution (via execution of a model update task) results in (1) a split of *Op1* into a two operation sequence *(Op1 Op1*)* reflecting completed and uncompleted portions of the original *Op1*, (2) allocation of a "machine repair" operation for the designated interval, and (3) subsequent detection and posting of two capacity conflicts. Because both conflicts involve the same resource, they are aggregated for simultaneous consideration. Analysis of this aggregate event indicates high resource contention over the conflict horizon, positive projected lateness in the jobs involved in the conflict, and a large conflict size, leading to formulation of an RSC revision task. Given that resource contention is lower at the aggregate *WA1* level in the hierarchical model (suggesting resequencing flexibility if alternative machine assignments are considered), *WA1* is chosen as the focal point of the RSC task. The revisions made by the RSC in this step are indicated in Figure 12. The resulting solution state is given in Figure 13.

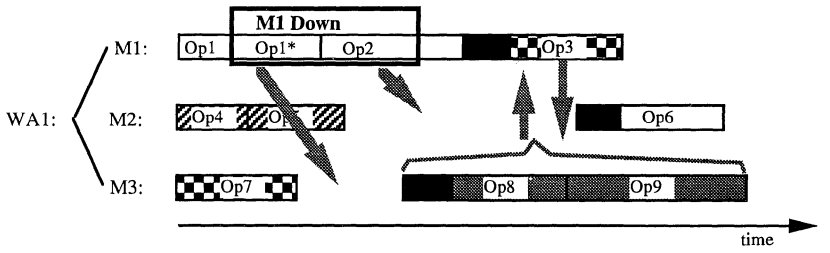


Figure 12. Action of RSC in revising *WA1* schedule

Execution of the RSC scheduling task introduces two new time conflicts, reflecting precedence violations between operation pairs *(Op1* Op1b)* and *(Op2 Op2b)* respectively. On the basis of conflict urgency, problem selection determines an initial focus on the conflict involving Job1. Analysis of this conflict indicates relatively low resource contention on *M4* over the conflict horizon, but positive projected lateness in the job involved in the conflict. Accordingly, a PV-OSC scheduling task is formulated to provide an aggressive approach to revising the downstream portion of Job1's sched-

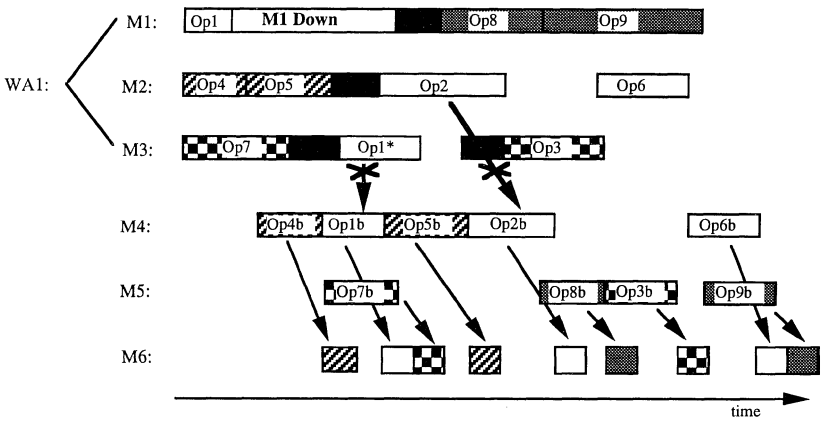


Figure 13. Solution State after revision of *WA1* schedule

ule. Figure 14 indicates the result of this action. Because weighted tardiness prioritization favors Job1 over both Job5 and Job2, decisions are made to utilize machine capacity currently allocated to both of these jobs. Upon commitment to these decisions, the depicted capacity conflicts are detected and posted. The previously detected precedence violation between the operation pair (*Op2Op2b*) persists also (the earliest feasible start time of *Op2b* is indicated by the "[" in Figure 14).

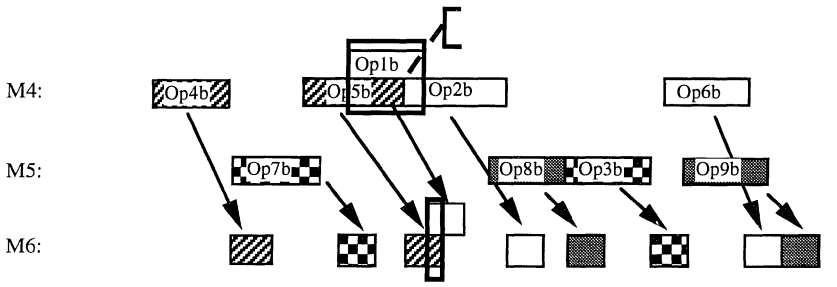


Figure 14. Scheduling state after PV-OSC (init-op = Op1b)

In this case, commonality in the jobs involved in each capacity con-

flict results in creation of an aggregate conflict event. Subsequent conflict analysis again indicates relatively low contention at resource focal points as well as the presence of rescheduling flexibility in the form of upstream slack. A CV-OSC scheduling task is formulated to reschedule the downstream operations of Job5 from *Op5b* forward using existing intervals of available capacity. The results of this action are indicated in Figure 15.

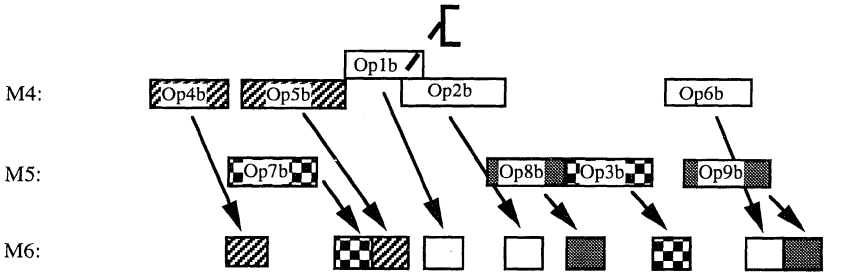


Figure 15. Solution state after CV-OSC (init-op = Op5b)

The final remaining conflict is also resolved using CV-OSC, because of the absence of significant downstream resource contention and negative projected job lateness. The final revised schedule is shown in Figure 16.

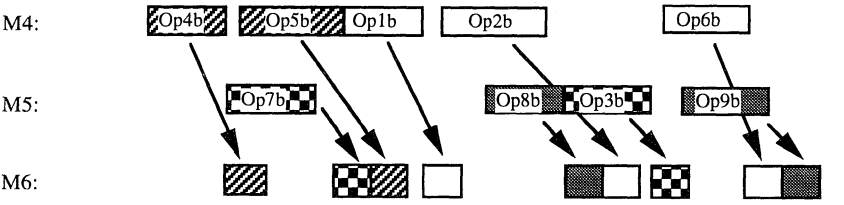


Figure 16. Final solution

5.4. Characteristics of the Approach

The OPIS scheduling methodology makes several pragmatic assumptions to deal with the special characteristics and complexities of the schedule modification task pointed out in Section 2. Each of these assumptions

involves compromise in some respect, but it is also these assumptions that make the approach viable in practical scheduling domains. In this section, we examine these assumptions in relation to the issues previously raised.

Bounding the scope of a given reaction. On each cycle of the repair process, OPIS makes heuristic decisions as to the scope of the next schedule modification action. These decisions are based on implications of the structure of current solution constraints as encoded in its subproblem formulation model. However, the conflict analysis metrics on which these implications are founded are, by computational necessity, aggregate and interpreted qualitatively. There is no guarantee that solution of the delineated subproblem (which limits visibility to a subset of the current solution constraints) will not introduce additional constraint conflicts into the schedule, and there are accordingly no a priori bounds that can be placed on the extent to which change will ripple through the solution. The expectation is that, on average, the heuristic model will limit the amount of revision performed to what is required (given expected performance biases - see below). Experimentation has borne out this expectation. Although missed opportunities for more localized change have been observed in specific reactive circumstances (e.g., when the current state falls at the boundaries of the model), overall results indicate that, on average, this model significantly outperforms other, more rigid and less informed schedule revision strategies [25]. Moreover, because the model is based on generic properties of the underlying solution constraint graph, it is applicable across a range of scheduling domains (and certainly generalizable to others).

Trading off attending to scheduling objectives for being non-disruptive. The degree of emphasis given toward either of these concerns is also a function of the heuristic subproblem formulation model employed. Abstract characterizations of the current solution state have implications with respect to both optimization needs and opportunities for non-disruptive modification, and these implications are not always synergistic. In such circumstances, the current OPIS subproblem formulation model is biased toward modification actions that preserve schedule quality. Other models are of course possible. For example, within the NEGOPRO software project scheduler [23], which adopts a similar constraint-directed, multi-perspective approach, a subproblem formulation model biased in the opposite direction is employed (in this domain, continuity of resource assignments is preferred even at the expense of acquiring more resources when milestones slip).

"Time to execution" bias. Although the above tradeoff has implications with respect to computational efficiency, the iterative nature of the modification process allows the need for responsiveness in reactive situations to be treated separately. On any given cycle of the iterative process, constraint propagation produces descriptions of the currently pending set of constraint conflicts and improvement opportunities. If prioritization of this set is based first on the temporal proximity of these problems to the current time then it can be ensured that conflicts blocking execution are

attended to in bounded time. Moreover, because new external updates are imported on each cycle, focus can shift to more pressing problems as they arise.

Guarantee of a feasible solution. The existence of a feasible solution is guaranteed in any modification context by assuming that constraints are "infinitely relaxable" along at least one dimension (e.g., activities can always be delayed, additional resource capacity can always be acquired, processes can always be dropped from the schedule). This flexibility is counterbalanced by the inclusion of corresponding scheduling objectives to minimize relaxation to the extent possible (e.g., minimize tardiness), and the subproblem formulation model's overall bias toward maintaining solution quality.

6. Relationship to Other Work

Work in reactive scheduling and schedule revision techniques has gained considerable prominence within the field of knowledge-based scheduling in recent years. Relative to manufacturing scheduling, several alternative constraint-based approaches have appeared. In [5], a blackboard-based system for generating and revising factory schedules based fairly directly on the concepts of OPIS (although employing different revision operators) is described. One interesting aspect of this work, from an architectural perspective, was its use of a decomposable framework for temporal constraint propagation that provided a basis for explicitly controlling the amount of constraint propagation performed in different problem solving circumstances, and thus an ability to trade off the time spent revising the schedule against the quality of the reaction.

In [4], a distributed constraint satisfaction problem solving (CSP) approach to reactive scheduling based on a hierarchically organized set of scheduling agents is defined. Each agent is given responsibility for making and retracting specific commitments (time assignments on a particular machine, machine assignments in a given work area, due dates of current jobs). Schedule revision proceeds as a collective activity based on dependency-directed backtracking search, with the single strategic agent bearing responsibility trading off global objectives when it has been established that current constraints cannot be met. From a behavioral standpoint, the interaction protocols employed give rise to a reactive strategy where scope is systematically enlarged from individual machine schedules, to groups of substitutable machines, and finally to consideration of process due date relaxation. The real issue here is scalability, given the reliance on systematic backtrack search.

Recent work in iterative repair scheduling approaches [3, 13, 16, 33] shares the same incomplete search assumption as OPIS in approaching conflict resolution. However, this work has principally focused on a different class of scheduling problems (e.g., space mission scheduling) where absolute

temporal constraints are not relaxable and there is generally less temporal structure to processes. The optimization problem here can be seen as minimizing the number of capacity conflicts (because any conflicts that cannot be resolved will require processes to be dropped from the schedule), and thus the tradeoffs emphasized in the OPIS reactive model concerning tardiness and WIP minimization are not relevant. One exception is the space shuttle processing domain of [33], which presents a complex "project scheduling" problem that is much closer in character to manufacturing scheduling problems. However, the approach taken here treats due dates as non-relaxable and attempts to minimize the number of capacity conflicts in the final schedule.

Work in iterative repair scheduling also differs in the approach taken to control of the revision process. Whereas OPIS relies on knowledge about problem structure to direct application of sophisticated local search methods, most of this work has instead emphasized more extensive global search with simpler (typically smaller granularity) repair heuristics. In [13, 16], a "min-conflict" heuristic, which revises a single conflicting commitment with bias toward minimizing the number of new conflicts introduced, is repeatedly applied to attempt to find a conflict free schedule from an initial randomly generated set of commitments (using periodic restarts as a hedge against local minima traps). In [33], a more complex heuristic that ensures continued consistency of relative temporal process constraints while it moves a conflicting operation is repeatedly applied within a simulated annealing search framework. In this case, provisions are provided for introduction of additional repair heuristics to deal with constraint violations other than capacity conflicts, and utility-based penalty functions associated with constraints and objectives are used to evaluate alternative solutions. The approach taken in [3] is perhaps the closest in concept to the OPIS methodology. Here extensive initial global search, which randomly shuffles activities to other regions on the time line to relieve capacity conflicts, is used as a basis for discovering inherent bottleneck regions in the space and this information is used to direct the application of more informed schedule revision operators.

7. Extensions and Current Focus

Subsequent research in manufacturing scheduling domains has explored frameworks for distributing and refining variants of the scheduling methodology employed by OPIS. In [28, 12], a structural decomposition of the factory, employing hierarchical descriptions of time and resource capacity constraints, was used as a basis for extending the OPIS framework to manage schedules at different levels over different time horizons. The CSS scheduler [21], alternatively explored a decomposition of scheduling responsibility among a set of "resource broker agents" (each concerned with efficient utilization of a specific set of resources) and a single "work order

manager" agent (concerned with process oriented constraints and objectives). In [24], a framework for partitioning responsibility between a global scheduler and a set of local "dispatching" agents was defined, emphasizing exploitation of existing temporal flexibility in the global schedule as a means for localized, execution-time schedule repair as circumstances warrant. This framework is currently being further developed for operational use in real-time control of an INTEL wafer fabrication facility.

One area of current research is investigating the broader applicability of these results and the basic OPIS schedule repair methodology to the problem of distributed management of large-scale transportation schedules. Work here has led to development of the DITOPS (DIstributed Transportation Scheduling in OPIS) system [32], which is currently being applied to problems in military crisis-action deployment logistics. Generalizations and extensions have been made to the original OPIS representational framework, the repertoire of revision methods available, and the heuristic subproblem formulation model to better reflect the character of the important constraints in this domain and the dimensions along which constraint relaxation and schedule modification should proceed. We have demonstrated capabilities to intelligently respond to a range of reactive problems (e.g., port closings, unexpected unavailability of transport vehicles, changes to material movement requirements), and are currently engaged in integrating DITOPS with other, complementary transportation planning technologies to support a large-scale demonstration in an operational scenario.

A second thrust of current research focuses on integration of incremental, reactive scheduling techniques with user decision-making processes, and the development of flexible interactive scheduling tools. As we argued at the outset of this paper, scheduling in most practical environments is very much an iterative process of "getting the constraints right". Incremental schedule revision techniques such as those developed within the OPIS scheduler are particularly well suited to the form of decision support that this process implies, particularly in substantial scheduling environments (manufacturing-related or otherwise) where it is unreasonable to expect users to productively interact with a scheduling system at the level of individual decisions in the system's solution model. In these domains, there are simply too many decisions to consider and effectively manipulate on an individual basis (e.g., via manual Gantt chart editing with constraint checking), and the details of most decisions are generally unimportant from the standpoint of user tasks and goals. The reactive scheduling methodology of OPIS, alternatively, provides the foundation for an interactive framework that preserves the user's ability to exercise explicit control over the solution change process (essential for exploration, understanding and calibration of possible solution improvements and adjustments) while promoting user interaction in higher-level and more comprehensible task-oriented terms (e.g., reschedule job x to complete by Friday, add sufficient overtime to complete next week's orders on time). Building on the notions of hierarchical model-

ing and opportunistic subproblem formulation defined in the OPIS control architecture, we are developing an interactive scheduling framework that operationalizes this view. Our vision is a graphical, spreadsheet-like model of user/system interaction, where schedules are visualized, analyzed and manipulated from aggregate decision-making perspectives, and the system incrementally "manages the details" of solution change in a manner consistent with user expectations. [31].

A third complementary component of our current research aims at simplification of the application building process. Though we believe that reactive scheduling methodology developed in OPIS is relevant to a broad range of applications, different scheduling environments invariably present different challenges. There is diversity along several dimensions: in the structure of different domains (e.g., type of manufacturing system and discipline), in the types of constraints that dominate, in the performance objectives and preferences that must be attended to, and in the types of uncertainties must be accommodated; and problem characteristics along each of these dimensions will dictate the modeling assumptions, scheduling heuristics, and solution procedures most appropriate for successful application. The OPIS scheduling architecture provides a modeling and scheduling system infra-structure for encoding and integrating application-specific solution components. However, at the software level, its reliance on frame-based implementation techniques provide only limited support for encapsulation of component functionality and exploitation of a layered system semantics. We have recently completed an initial redesign and reimplementation of the basic OPIS framework, based on more modern object-based analysis and programming techniques. Our goal here is a application building "toolbox", which provides an a compositional library of system building blocks (e.g., constraint representation and propagation techniques, (re)scheduling methods and heuristics, subproblem formulation policies) and an explicit hierarchy of protocols for configuring and customizing schedulers to meet the requirements of specific applications.[30]

A final area of current research is exploring the development and use of constraint-based schedule revision strategies with more flexible "temporal data base" representations of current solution constraints (e.g., [6]). Within such representational frameworks, start and end times of scheduled activities are generally not fixed to specific points in time, but instead are constrained to intervals that satisfy posted temporal constraints. This allows, for example, the possibility of developing schedules that anticipate executional uncertainty and defer commitment on timing details whenever possible and appropriate. This work is being carried out with the HSTS system[18], an integrated planning and scheduling architecture that provides such a representational framework.

8. Acknowledgments

The OPIS scheduler is the result of the contributions of many individuals spanning a period of many years. Peng Si Ow provided inspiration for many of the ideas underlying the approach and was a research partner throughout the original system development. Discussions with Mark Fox have also had a major influence in shaping the OPIS approach. Other individuals who have made valuable contributions include Gilad Amiri, Dina Berkowitz, Casper Cheng, Geir Hasle, Juha Hynynen, Ora Lassila, Claude LePape, Jyi-Shane Liu, Dirk Matthys, Bruce McLaren, Pedro Muro, Nicola Muscettola, Kevin Neel, Kikuo Ogaki, Jean-Yves Potvin, Ali Safavi, Toshihiro Satomi, Katia Sycara, and Christopher Young.

This research has been sponsored in part by the Air Force Office of Scientific Research under contract F49620-82-K-0017, International Business Machines Inc. under contract number 71223046, the Advanced Research Projects Agency under contract F30602-90-C-0119 and the CMU Robotics Institute.

An earlier version of this paper appeared as "OPIS: A Methodology and Architecture for Reactive Scheduling" in *Intelligent Scheduling*, (eds. M. Zweben and M.S. Fox), Morgan Kaufmann Publishers, San Mateo, CA, 1994.

REFERENCES

1. Amiri, G. and S.F. Smith, "A Comparative Analysis of Constraint-Based Scheduling Strategies", The Robotics Institute, Carnegie Mellon University, Tech. Report, September, 1992.
2. Bean, J.C., and J.R. Birge, "Match-Up Real-Time Scheduling", Technical Report No. 85-22, University of Michigan, Department of Industrial and Operations Engineering, June, 1985.
3. Biefeld, E. and L. Cooper, "Operations Mission Planner: Final Report", Jet Propulsion Laboratory, Tech. Report 90-16, March, 1990.
4. Burke, P., and P. Prosser, "A Distributed Asynchronous System for Predictive and Reactive Scheduling", Tech. Report AISL42, Dept. of Computer Science, University of Strathclyde, 1989.
5. Collinot, A., C. LePape, and G. Pinoteau, "SONIA: A Knowledge-Based Scheduling System", *Artificial Intelligence in Engineering*, 2 (2), 1988, pp. 86-94.
6. Dean,T.L. and D.V. McDermott, "Temporal Data Base Management", *Artificial Intelligence*, 32 (1), April, 1987, pp. 1-56.
7. Erman, L.D., F. Hayes-Roth, V.R. Lesser and D.R. Reddy, "The Hearsay-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, 1980.
8. Fox, M.S., "Constraint-Directed Search: A Case Study in Job Shop Scheduling", PhD. Thesis, Dept. of Computer Science, Carnegie Mellon University, 1983.

9. Fox, M.S., and S.F. Smith, "ISIS: A Knowledge-Based System for Factory Scheduling", *Expert Systems*, 1 (1), July, 1984, pp. 25-49.
10. Fox, M.S., N. Sadeh, and C. Baycan, "Constrained Heuristic Search", *Proceedings IJCAI-89*, Detroit, MI, August, 1989.
11. Graves, S.C., "A Review of Production Scheduling", *Operations Research*, 29 (4), 1981, pp. 646-675.
12. Hynynen, J., "A Framework for Coordination in Distributed Production Management", Ph.D. Thesis, Helsinki University of Technology, Laboratory of Information Processing Science, October, 1988.
13. Johnston, M., "SPIKE: AI Scheduling for NASA's Hubble Space Telescope", *Proceedings 6th IEEE Conference on Artificial Intelligence Applications*, Santa Barbara, CA, March, 1990, pp. 184-190.
14. LePape, C. and S.F. Smith, "Management of Temporal Constraints for Factory Scheduling", Technical Report TR-CMU-RI-87-13, The Robotics Institute, Carnegie Mellon University, June, 1987.
15. Meleton, M.P., "OPT: Fantasy or Breakthrough", *Production and Inventory Management*, Second Quarter, 1986, pp. 13-21.
16. Minton, S., M.D. Johnston, A.B. Phillips and P. Laird, "Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method", *Proceedings AAAI-90*, Boston, July, 1990.
17. Johnson, L.A. and D.C. Montgomery, *Operations Research in Production Planning, Scheduling and Inventory Control*, John Wiley, New York, 1974.
18. Muscettola, N., S.F. Smith, A. Cesta, and D. D'Aloisi, "Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Framework", *IEEE Control Systems*, Vol. 12, No. 1 (February, 1992).
19. Ow, P.S.,"Focused Scheduling in Proportionate Flowshops", *Management Science*, 31 (7), July, 1985, pp. 852-869.
20. Ow, P.S., S.F. Smith and A. Thiriez, "Reactive Plan Revision", *Proceedings AAAI-88*, St. Paul, MN, August, 1988.
21. Ow, P.S., S.F. Smith and R.E. Howie, "CSS: A Cooperative Scheduling System", in *Expert Systems and Intelligent Manufacturing*, (ed.) M.D. Oliff, Elsevier Science Publishing Co., 1988.
22. Ow, P.S. and S.F. Smith, "Viewing Scheduling as an Opportunistic Problem Solving Process", in *Annals of Operation Research*, Vol. 12, (ed.) R. Jareslow, Baltzer Scientific Publishing Co., 1988.
23. Safavi, A. and S.F. Smith, "A Heuristic Search Approach to Planning and Scheduling of Software Projects", in *Advances in Artificial Intelligence: Natural Language, and Knowledge-Based Systems*, (ed.) M. Golumbic, Springer-Verlag Publishers, 1990.
24. Smith, S.F., N. Keng, and K. Kempf, "Exploiting Local Flexibility During Execution of Pre-Computed Schedules", in *Applications of AI in Manufacturing* (eds. D. Nau and C. Tong), MIT Press, July, 1992.
25. Smith, S.F., P.S. Ow, N. Muscettola, J.Y. Potvin, and D. Matthys, "An Integrated Framework for Generating and Revising Factory Schedules",

Journal of the Operational Research Society, 41(6), June, 1990.

26. Smith, S.F., "The OPIS Framework for Modeling Manufacturing Systems", Technical Report TR-CMU-RI-89-30, The Robotics Institute, Carnegie Mellon University, December, 1989.
27. Smith, S.F., "A Constraint-Based Framework for Reactive Management of Factory Schedules", in *Intelligent Manufacturing*, (ed.) M.D. Oliff, Benjamin Cummings Publishers, 1987.
28. Smith, S.F. and J.E. Hynynen, "Integrated Decentralization of Production Management: An Approach for Factory Scheduling", in *Intelligent and Integrated Manufacturing Analysis and Synthesis*, (eds.) C.R. Liu, A. Requicha, and S. Chandrasekar, ASME PED-Vol. 25, New York, 1987.
29. Smith, S.F. and P.S. Ow, "The Use of Multiple Problem Decompositions in Time-Constrained Planning Tasks", *Proceedings IJCAI-85*, Los Angeles, CA, August, 1985.
30. Smith, S.F. and O. Lassila, "Configurable Systems for Reactive Production Management", in *Knowledge-Based Reactive Scheduling*, (eds.) R. Kerr and K. Szelke, IFIP Transactions 15-B, North Holland Publishers, Amsterdam, 1994.
31. Smith, S.F. and O. Lassila, "Toward the Development of Flexible Tools for Mixed-Initiative Transportation Scheduling", *Proceedings 1994 ARPA Planning Workshop*, Tuscon, AR, February, 1994.
32. Smith, S.F. and K.P. Sycara, "Distributed, Multi-Level Management of Transportation Schedules", Technical Report CMU-RI-TR-93-17, The Robotics Institute, Carnegie Mellon University, December, 1993.
33. Zweben, M., M. Deale, and R. Gargan, "Anytime Rescheduling", *Proceedings 1990 DARPA Workshop on Innovative Approaches to Planning, Scheduling and Control*, (ed.) K.P. Sycara, Morgan Kaufmann Publishers, 1990.

Intelligent Scheduling with Machine Learning

SANG C. PARK[1], SELWYN PIRAMUTHU[2], NARAYAN RAMAN[3], and MICHAEL J. SHAW[4]

[1]Graduate School of Business, University of Wisconsin, Madison, WI 53706, spark@bus.wisc.edu

[2]Department of Decision and Information Science, University of Florida, Gainsville, FL 32611, selwyn@nerix.nerdc,ufl,edu

[3]Department of Business Administration, University of Illinois, Champaign, IL 61820, raman@uiucvmd.bitnet

[4]Department of Business Administration, University of Illinois, Champaign, IL 61820 and Beckman Institute, University of Illinois, Urbana, IL 61801, mshaw@ux1.cso.uiuc.edu

Abstract

This paper presents a methodology for intelligent scheduling based on machine learning. Simulation is used to generate training examples depicting manufacturing patterns and the resulting performance of scheduling rules. Heuristics for applying these rules can then be generated by inductive learning algorithm. The methodology also incorporates mechanisms for rule refinement and meta-control. The approach is addressed in the context of the adaptive decision support system (DSS) managing the interactions between simulation, training, learning, and scheduling.

The resulting DSS, referred to as pattern directed scheduling system (PDS), has been applied to the flexible manufacturing system (FMS) and flexible flow system (FFS) environments. Computational results reveal that such a pattern directed scheduling approach results in superior system performance.

keywords: intelligent scheduling, machine learning, simulation, rule refinement, adaptive decision support system, flexible manufacturing system, and flexible flow system

1. Introduction

We consider both flexible manufacturing system (FMS) and flexible flow system (FFS) environments to develop the proposed adaptive intelligent scheduling system. FMSs have routing flexibility which enables them to adapt easily to changes in the operations that need to be done on consecutive jobs. FMSs also have buffer-size limitations which avoids build-up of incomplete jobs in the system. Most FMS studies that incorporate routing flexibility consider machine assignment at the system setup stage only, and could be considered to be stationary. During operation in real time, only one route is used. Such stationary schedules do not, however, consider the need for various scheduling decisions to be made in real time as production proceeds. As Buzacott (1982) notes, permitting machine assignment capability in real time can improve overall system performance. In this study, we permit both machine assignment and operations scheduling in real time, thus leading to real-time dynamic scheduling. Furthermore, in comparison to previous studies, we consider a wider range of system characteristics. The control variables used in this study are overall system utilization level, tightness and variability of due dates, relative machine workload balance, actual buffer size and the degree of routing flexibility.

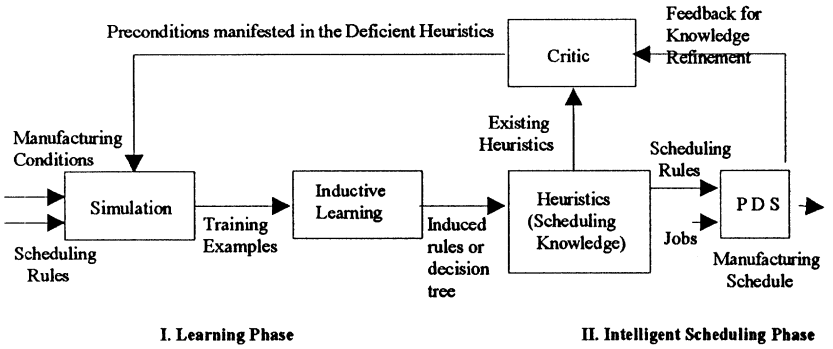
More importantly, however, this research focuses on a pattern directed heuristic scheduling process. The relative effectiveness of a given scheduling rule likely depends upon the system characteristics. In a dynamic system, these characteristics change over time. Therefore, it appears conceptually appealing to adopt an approach which employs appropriate and possibly different scheduling rules at various points in time. In order to do so, however, we need a mechanism which can distinguish different combinations of system characteristics, or patterns, from one another.

We propose a systematic methodology which provides a pattern directed scheduling (PDS) with machine learning capabilities in order to extract patterns from the dominant characteristics of a given scheduling environment. These patterns are subsequently incorporated into a set of heuristic rules which guide the selection of scheduling rules in real time. We employ the C4.5 (Quinlan 1988) a descendent of ID3 algorithm to help PDS recognize patterns comprising of the manufacturing system and job characteristics. These patterns are then used to specify the conditions which trigger the use of a given scheduling rule. These hybrids, which are basically heuristics in the form of (1) patterns as preconditions and, (2) the assignments of scheduling rules as the resulting actions, are then added to the PDS's knowledge base in the course of learning. The organization of the PDS is shown in Figure 1-1.

We also suggest an approach for applying the developed PDS framework for dynamic scheduling in a flexible flow system (FFS). The extension to flexible flow shop (Wittrock, 1985) scheduling involves two sets of scheduling rules (one for the part release and the other for the dispatching), which need to be controlled in turn by meta-level heuristic rules to maintain the integrity of the FFS system.

The advantages of the framework proposed in this study are: 1) an intelligent scheduling DSS with learning abilities, 2) capability of the DSS to be adaptive to changes in the environment of interest, and 3) automated knowledge acquisition and refinement through inductive learning.

Figure 1-1: Framework for Pattern-Directed Scheduling



The rest of this paper is organized as follows. Review on pattern directed scheduling system is briefly discussed in section 2. Section 3 presents some basic concepts of inductive learning and discussion on how inductive learning can be combined with simulation. In Section 4, we describe the application of PDS to the FMS. Dynamic scheduling in a FFS is next discussed in Section 5. Section 6 deals with the addition of a learning module to the traditional DSS, and how this aids in learning and knowledge refinement. We conclude in Section 7 with a summary discussion.

2. Review on Pattern Directed Scheduling System

A job scheduler for manufacturing systems has to consider various events (e.g. machine breakdowns) that might occur in the system randomly. This creates an instability in the proposed system if appropriate heuristics are not used in such a case. The performance of the entire system is improved by dynamically using heuristics consistent with the pattern which is exhibited in the system. The adaptive scheduling based on current patterns in the system helps take advantage of various heuristics when they are at their best in terms of performance. In this section, we discuss one such system called pattern directed scheduling system (PDS), which combines adaptive application of scheduling rules with learning.

The performance of PDS in FMS environment is first reported (Park et al. 1989), and recognized in the context of dynamic operations planning which

is concerned with the dynamic, minute-to-minute operations of FMS (Gunasekaran et al. 1993). Moreover, PDS is recognized as an intelligent scheduling system which has applied learning principles in manufacturing scheduling using a simulation experiment as an example or an observation (Pierreval 1992). The category of learning principles that PDS utilizes is an inductive learning (a type of learning from example) which generates decision trees (Holsapple et al. 1993; Chaturvedi et al. 1993).

Ever since, PDS has been successfully implemented in various types of manufacturing system environments including flexible manufacturing (FMS) systems and flexible flow systems (FFS). PDS proves itself to be effective in non stationary manufacturing environment. Such non stationary situations include but are not limited to sudden machine breakdown, job due date changes, urgent jobs rush-in, and cancellation of jobs. PDS's unique characteristic of being adaptiveness enables the state-dependent selection of best scheduling rule during the real time scheduling, thus results in not only the same but even better scheduling performance than any best scheduling rules does.

Extensive testing on the lab settings of the FMS reveals that PDS performance is equal to or better than that of the best scheduling rule in 91% of the testing cases under the stationary manufacturing environment (Park 1991). When compared to each single scheduling rules' average mean tardiness, that of the PDS is always the smallest (Park et al. 1993). PDS performance is much better than that of the best scheduling ruled in 92% of the testing cases under non stationary manufacturing environment (Park 1991). PDS has been implemented in an ancillary company that manufactures fuel delivery systems for passenger cars and light trucks. Two sets of experiments were conducted to evaluate the performance of the PDS to those of other scheduling rules under stationary and non stationary conditions, respectively. The results are consistent with overall superiority of PDS observed in earlier experiments (Park et al. 1993). Analysis on PDS performance over varying conditional factors of the manufacturing environment was reported in Shaw et al. 1992. The test-bed for the PDS has been extended to the FFS under the minimizing flow time objective in both FMS and FFS environment (Park et al. 1990; Piramuthu et al. 1993).

In this study, we will report the full-fledged implementation of PDS in the FFS along with the comparative study on its implementation in FMS under different scheduling criteria. We expect the shift of relative importance of control attributes.

3. Integrating Inductive Learning and Simulation

Both inductive learning and simulation, when integrated, are complementary to each other in learning by experimentation. As per the training examples generated using simulation, inductive learning is used to generate the heuristic rules which are then adaptively applied in the PDS environment. The learning module requires a source (simulation module) for training examples

which exemplifies the system under consideration, and the simulation module requires a source (inductive learning module) for learning heuristics. Thus there is a feed-back between the inductive learning and simulation modules which happens to enhance the performance characteristics of both the modules by knowledge refining thus increasing the performance of the PDS.

As the patterns in the system varies over time, the integration of inductive learning and simulation helps in adaptively utilizing different scheduling rules in real-time. Unlike in stationary systems where extensive calculations are required whenever there is an unexpected disturbance in the system such as machine breakdowns, and large deviations in processing times, the PDS incorporating both simulation and inductive learning adapts itself instantaneously. This is rendered possible through learning the different possible patterns that can arise in the system and their corresponding scheduling rules. Thus the decision support system incorporating simulation is used as a learning tool in addition to being used as an evaluating tool.

As in discrete-event simulation, a major disadvantage with this method, however, is that it does not eliminate the need for the decision maker to select the initial set of variables that are included in the model. The performance of the final model depends, to a large extent, on the input information as per the variables that are used in the system. In some cases, inductive learning can be used to signal a deficit in the variables that are used (not enough number of descriptors used). This happens when, at a leaf node, values of all the variables in the path of the decision tree are the same for more than one example, although they belong to different classes. This signals the simulation module to include new variable(s) for describing the training observations to distinguish between the classes of interest.

3.1 Inductive Learning

Inductive learning can be defined as the process of inferring the general description (that is, the concept) of a class from the description of individual observation in the training examples. A concept here is a conjunction of most discriminant descriptors which identifies a given class. The concept to be learned in FMS scheduling, for example, can be a description of manufacturing environments suitable for applying a given scheduling rule (a class).

A set of training examples are provided as input for learning the concept representing each class. A given training example consists of a vector of attribute values and the corresponding class. A concept learned can be described by a rule determined by inductive learning. If a new input data case satisfies the conditions of this rule, then it belongs to the corresponding class. For example, a rule defining a concept can be the following:

IF $(b_{i1} \geq a_{i1} \geq c_{i1})$ AND $(a_{i2} = d_{i2})$ AND ..... AND $(b_{im} \geq a_{im} \geq c_{im})$ THEN τ

where a_{ij} represents the jth attribute in $Rule_i$, and b_{ij} and c_{ij} define the range for

a_{ij} where as d_{ij} represents a nominal (symbolic) value. In the FMS scheduling problem described in this paper, $Rule_i$ is treated as a selection heuristic with a conjunction of attribute conditions collectively defining the pattern, and τ represents the best scheduling rule for that pattern.

An instance that satisfies the definition of a given concept is called a positive example of that concept; whereas an instance that does not is a negative example of that concept. In the dynamic scheduling problem, since there are several scheduling rules which can potentially be selected, multiple concepts need to be learned. In this situation, the training examples supporting the use of a scheduling rule are treated as positive examples of that rule, whereas training examples supporting all the other scheduling rules are treated as negative examples.

A concept learned is an induced or generalized description of all supporting positive examples. For a set of training examples, the generalization process identifies the common features of these examples which differentiate (discriminate) them from the negative examples. The input to an inductive learning algorithm consists of three parts: 1) A set of positive and negative examples, 2) a set of generalization and other transformation rules, and 3) criteria for a successful inference. Each training example consists of two components -- a data case consisting of a set of attributes, each with an assigned value; and the classification decision made by a domain expert according to the given data case. The output generated by this inductive learning algorithm is a set of decision rules consisting of inductive concept definition for each of the classes. Learning programs falling into this category include AQ15 (Hong et al. 1986), PLS (Rendell 1986), ID3 (Quinlan 1986), and C4.5 (Quinlan 1988). These programs are sometimes referred to as "similarity-based" methods, as opposed to "explanation-based" methods (Mitchell et al. 1986).

Besides these similarity based learning methodologies, there is a methodology called CART (Classification and Regression Trees) in the field of mathematics and statistics (Breiman et al. 1984). As in the case of C4.5 and ID3, CART constructs a binary decision tree. It splits training examples by selecting the attribute a_i and its threshold value b_i that gives the best split CART tries to minimize the sample variance of the decisions (classes encoded 0 for the positive example and 1 for the negative example) of the spliced subset where as ID3 and C4.5 selects the attribute and its threshold value which yields the maximum information gain based on enthrophy theory. A disadvantage of CART is its basic assumption that individual variables are guaranteed to be normally distributed.

In this section, we give an overview of the proposed methodology for incorporating inductive learning into the dynamic scheduling process. Our approach integrates simulation, inductive learning and pattern directed scheduling. It consists of the following three steps: 1) Generating training examples, 2) learning rules from examples, and 3) executing pattern directed scheduling. Detailed descriptions of these steps are in order.

3.2 Generating Training Examples

The major objective of generating training examples is to provide the correct characterization of the manufacturing environments suitable for various scheduling rules. We use simulation experiments to generate a variety of such environments. The relevant design issues which need to be addressed are:

Criteria for performance evaluation: These could be stated as scheduling objectives such as minimizing mean tardiness, minimizing mean flow time, etc. *Set of scheduling rules*: In order for the learning process to be comprehensive, we need to identify all scheduling rules which can potentially be appropriate for the scheduling objective studied.

Specification of control attributes: We need to determine the control attributes which are likely to be dominant for the purpose of describing a pattern, and which can, therefore, affect the selection of a given scheduling rule. These control attributes include both system and job characteristics.

Measurement of control attributes: There are several possible ways in which the control attributes discussed above can be measured in training examples for providing descriptions of the scheduling environment at various points in time. If the decision (the selection of a given scheduling rule) is based upon the cumulative average measure (that is, from time zero to the current time T_{now}), training examples must be expressed in terms of cumulative average values. However, if the decision is made on the basis of the anticipated values (projection at time k through time k + δ), training examples must do likewise. The cumulative average converges to the experimental design parameters, hence, its value is stable in the steady state. On the other hand, anticipated values show wide fluctuations and greater "lumpiness".

Training examples are generated by simulating the system performance for the desired scheduling objective under a variety of patterns. For a given pattern, the simulation runs are replicated for each scheduling rule considered. The one rule which results in the best performance yields the positive example for that pattern. In this manner, we are able to establish a one to one correspondence between a manufacturing pattern and the scheduling rule appropriate for that pattern.

3.3 Learning Rules from Examples

The inductive learning mechanism is used to establish formally the correspondence between the manufacturing patterns and the best scheduling rule for each pattern, as observed from the training examples. Using an automated inductive learner has several advantages: 1) qualitative information with nominal values can be analyzed together with quantitative information; 2) hypotheses can

be conditionally accepted; 3) the dominant aspects of design alternatives are systematically detected which can be used iteratively for redesigning the experiment; and 4) new concepts or knowledge in the form of heuristics can be created. This enables the computer to be a knowledge generator instead of being merely an information collector.

Moreover, the simulation mechanism complements the inductive learner during the process of heuristic modification. Inductive learning mechanisms used in the literature passively analyze given training example(s); they cannot control these training examples by themselves. Similarly, simulation has been used primarily for evaluation rather than for learning. By incorporating the simulation mechanism with the learning process in our methodology, the inductive learner can determine the number and the nature of training examples studied since it actively controls the generation of future training examples which are critically important for further performance of the system. Because of this feedback, the inductive learner is transformed from being purely data-driven to being model-driven. Therefore, not only is the burden of collecting off-line training examples greatly alleviated, but the time required to collect examples is shortened. More importantly, this approach automates the acquisition of knowledge for the manufacturing knowledge base. This integration of inductive learning and simulation provides an appealing framework for automated manufacturing decision support which learns through experimentation.

3.4 Executing Pattern Directed Scheduling

Pattern directed scheduling employs the learned heuristics for scheduling in real time (Shaw 1988, 1989). Whenever a scheduling decision is to be made, the pattern of the current state is observed. Once the pattern is recognized, it is compared with the preconditions of the developed heuristics. The matching heuristic is initiated and its resulting action (the appropriate scheduling rule) is used for assigning priorities to the waiting jobs.

PDS employs a filtering mechanism to prevent overreaction to changes in the scheduling environment. This mechanism maintains a cumulative score of the number of times a given scheduling rule is favored by the current pattern. A switch to this rule occurs only when this score exceeds a pre-specified threshold. In the following section, we discuss the various filtering mechanisms and their impact on the performance of PDS. (For further details, the reader is referred to Shaw, Park and Raman 1992).

4. Dynamic FMS Scheduling

An example of PDS application to the FMS scheduling is discussed in this section. FMS environment, with its ever changing scenarios, is an excellent domain for demonstration PDS's adaptiveness. Since the jobs coming into the system have disparate routings, there is a definite need to apply appropriate

scheduling rules as per the current status of the system.

4.1 Problem Specification

Scheduling Criterion

The scheduling objectives considered in this study are minimizing mean flow time and mean tardiness. These objectives were selected primarily because they have been studied extensively, and various scheduling rules have been found to perform well under different scheduling environments. This provides a strong benchmark for testing the effectiveness of the pattern directed scheduling approach.

Set of Scheduling Rules Considered

The scheduling rules used for the purpose of generating the PDS for the minimum tardiness objective are: 1) Earliest Due Date (EDD) rule, 2) Shortest Imminent Processing Time (SIPT) rule, 3) Modified Job Due Date (MDD) rule, 4) Modified Operation Due Date (MOD) rule, and 5) Critical Ratio (CR) rule. For the minimum flow time criterion, we used 1) SIPT, 2) Least Work Remaining (LWKR), and 3)the Anticipated Work in Next Queue (AWINQ) rules. These scheduling rules rank the various jobs competing for the use of a given machine at any point in time according to different priority schemes. They compute priority indexes for individual jobs; the job with the smallest index value is selected first. The priority index corresponding to each rule is given below:

EDD	d_j
SIPT	p_{ij}
MDD	$max\{t+P_{ij}, d_j\}$
MOD	$max\{t+p_{ij}, d_{ij}\}$
CR	$(d_j-t)/P_{ij}$
LWKR	P_{ij}

where d_j is the due date of job j; p_{ij} and d_{ij} are, respectively, the processing time and due date of operation i in job j; P_{ij} is the processing time remaining in job j at the start of operation i, and t is the time at which the scheduling decision is to be made. The AWINQ rule favors a job which is processed next on the machine which has a queue with the least anticipated work waiting. These rules have been found to be effective for the objective of minimizing mean tardiness or mean flow-time in the past (see, for example, Baker 1984).

Control Attributes Considered

We addressed an FMS which permitted random, job shop-like flow of parts through the system. For generating training examples, nine control attributes were considered: 1) number of machines (NMAC), 2) buffer size of the individual machine (TBF), 3) machine homogeneity (MH), 4) relative

machine workload (RLOAD), 5) bottleneck machine workload (BOTTLE), 6) blocking status (BLOCK), 7) contention factor (CFACT), 8) flow allowance factor (FA), and 9) overall system utilization (SU). Machine homogeneity refers to the similarity among machines. It is expressed as the ratio of the standard deviation of the number of operations that each machine can process to the average number of operations that a machine can process. Relative machine workload is expressed as the ratio of the standard deviation of the machine utilizations to the average machine utilization. Bottleck machine workload measures the ratio of the maximum workload at any machine to the average workload. Blocking status is a logical yes-no variable which indicates whether any machine in the system is being blocked at a given point in time. Flow allowance factor controls the tightness of job due dates. We used the following expressions to determine job due dates:

$$d_j = a_j + F * p_j$$

where a_j and p_j denote, respectively, the arrival time and processing time of job j, and F is the flow allowance factor.

4.2 Rule Induction

Based on the training examples, an inductive learning program, C4.5 (Quinlan, 1988), was employed to describe the patterns for selecting each scheduling rule. The rule induction process carried out by C4.5 results in the decision tree which was subsequently translated into a set of pattern directed heuristics for controlling the pattern directed scheduler.

The decision tree generated highlights the relative importance of the various attributes in characterizing the manufacturing environment; that is, it helps determine the set of most relevant attributes that comprise the patterns of the rules. Moreover, C4.5 selects the attributes to be included in the rules in such a fashion that the earlier an attribute is selected, the more important that attribute is in describing the patterns of the rule. An example of a typical decision tree generated for the minimum flow time criterion, and the heuristic rules derived from this decision tree are given in figures 4-1 and 4-2. Based on the learning results shown in Figures 4-1 and 4-2, the attributes that need to be considered for selecting scheduling rules can be ordered by their relative importance as follows: 1) NSDRL, 2) CFACT, and 3) TBF.

The decision tree thus derived is very informative. In figure 4-1, NSDRL, which is the standard deviation of relative loading measuring the degree of balance of the system, is closer to 1 if the system is balanced (workload of each machine is even), is chosen as the most important (most discriminant) attribute based on its information content used to distinguish between the different scheduling rules. SIPT, which is a myopic heuristic, consistently appears on the balanced side of the tree (when each machine workload is balanced) confirming the fact that SIPT performs well when the

system is balanced than otherwise. Similarly, AWINQ, which is a global heuristic since it takes into account work in the next queue also, performs better when the system is relatively unbalanced.

Figure 4-1: Decision Tree Under Min. Mean Flow time Objective

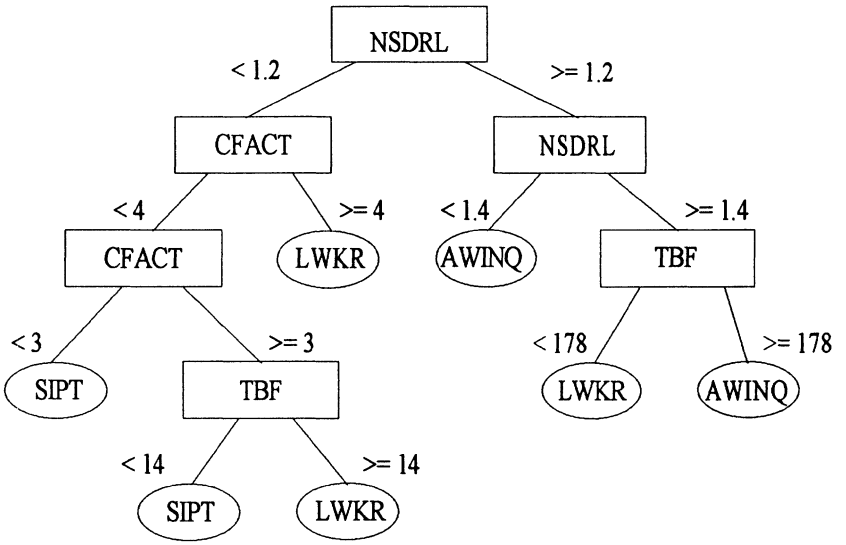


Figure 4-2: Heuristic Rules Derived From Figure 4-1

IF (NSDRL < 1.2) AND (CFACT < 4) AND (CFACT < 3) THEN SIPT
IF (NSDRL < 1.2) AND (CFACT < 4) AND (CFACT ≥ 3) AND (TBF < 14) THEN SIPT
IF (NSDRL < 1.2) AND (CFACT < 4) AND (CFACT ≥ 3) AND (TBF ≥ 14) THEN LWKR
IF (NSDRL < 1.2) AND (CFACT ≥ 4) THEN LWKR
IF (NSDRL ≥ 1.2) AND (NSDRL < 1.4) THEN AWINQ
IF (NSDRL ≥ 1.2) AND (NSDRL ≥ 1.4) AND (TBF < 178) THEN LWKR
IF (NSDRL ≥ 1.2) AND (NSDRL ≥ 1.4) AND (TBF ≥ 178) THEN AWINQ

On the contrary, the set of most important attributes in the decision tree generated under the minimum mean tardiness criterion differs from the one selected under the minimum mean flow time criterion. They are contention factor, relative loading, overall system utilization, flow allowance factor, and

bottleneck machine workload. An example heuristic rules generated for the minimum tardiness criterion is given in figure 4-3. Again SIPT works well under the highly balanced job shop like system (CFACT ≤ 2.139 & NSDRL ≤ 0.298). For the typical FMS environment where more than 2 machines are available for processing each operation (CFACT > 2.139),MOD works well when the system is not congested (overall system utilization level is less than 72%), otherwise EDD turns out to be the best.

Figure 4-3: Heuristic Rules Under Min. Mean Tardiness Objective

IF(CFACT ≤ 2.139) AND (NSDRL ≤ 0.298) THEN SIPT
IF(CFACT ≤ 2.139) AND (NSDRL > 0.298) AND (SU ≤ 64.0735) THEN MDD
IF(CFACT ≤ 2.139) AND (NSDRL > 0.298) AND (SU > 64.0735) AND (FA > 4.0) THEN MDD
IF(CFACT ≤ 2.139) AND (NSDRL > 0.298) AND (SU > 64.0735) AND (FA ≤ 4.0) THEN EDD
IF (CFACT > 2.139) AND (SU > 71.2341) THEN EDD
IF (CFACT > 2.139) AND (SU ≤ 71.2341) THEN MOD

4.3 Pattern Directed Scheduling

Decision heuristics similar to those in Figure 4-2 provide the basis for pattern directed scheduling. The left-hand-side of each heuristic represents the pattern for applying the scheduling rule at the right-hand-side. Thus, different scheduling rules will be selected when the manufacturing system manifests different patterns. As a result, rule selection decision can also be changed dynamically as the pattern of the system changes over time. Conceptually, this technique should be more effective than the use of a single rule throughout the process, as has been the case in most of the previous research on dynamic scheduling. We use simulation studies to provide experimental evidence in support of this conjecture for the minimum tardiness objective (Park 1991; Park et al. 1993).

The classification and prediction accuracy of the learned heuristics depends on the contents of the training examples, which are the sole input to the inductive learning module. Since in discrete-event simulation, only the average of the attribute values over time are taken into consideration, variation in the training examples to a certain degree is unavoidable. The rules that are developed using the inductive learning module thus needs to be filtered to reduce any noise that might be present due to averaging. This filtering is achieved by adjusting the reactiveness of the scheduling system to pattern changes.

In applying PDS, the issue of how reactive the scheduling system is to pattern changes turns out to be significant. Specifically, the performance of PDS

deteriorates when different rules are applied immediately upon a pattern change. This problem can be attributed to excessive system nervousness and over-reaction to patterns that are only transient. In order to mitigate this tendency, we used a threshold factor which smoothes the response of PDS to pattern changes. Under this approach, we keep a cumulative score of the number of times each rule is favored. Suppose that the scheduling rule being used currently is i^* with a cumulative score of S_{i^*}. If a pattern change now favors the selection of rule j, then j will be selected if and only if $S_j > R * S_{i^*}$. R is a smoothing constant which is treated as a variable. The appropriate value of R for a given pattern is determined through separate application of the inductive learning.

The performance of PDS depends upon (1) the frequency of actual pattern changes that are realized in the system, (2) system size, (3) contention factor, (4) mean tardiness values, and (5) performance differences of the various dispatching rules used. If the number of potential switches in the applicable scheduling rule is high, the frequency of actual pattern changes are regulated by the R values. The improvement ratio (between the performance of PDS and that of the best scheduling rule) decreases with an increase in the number of pattern changes (Shaw, Park, and Raman, 1992). On the other extreme, only a few pattern switches would imply that there are a few dominant patterns that characterize the system. Hence, PDS is most effective when the number of pattern changes is from medium to reasonably high and no pattern is clearly dominant

The differential between utilization of machines would tend to be lesser when there are more machines than are needed for jobs to be processed, and PDS would behave identically to the best dispatching rule, in the limit, as the contention factor is increased. An increase in the system size, as measured by the number of machines, generally leads to an increase in contention factor as well.

As in Baker (1984), when the mean tardiness values are small, the relative difference in the performance of PDS and best dispatching rule decreases. Under the conditions found in a FMS environment, if all the dispatching rules that are being used for PDS have similar performance, the difference between PDS and the best heuristic diminishes since, ideally, PDS can perform only as well as the best rule in a given situation.

In general, because the set of training examples is a subset of the universe of the possible scheduling environments, the learned heuristics are likely to be over generalized. Therefore, some prediction errors cannot be avoided. Specifically, in a situation which is not described in the set of training examples, PDS may not perform very well. However, we found that in such situations, the difference between the performance of PDS and the best rule was marginal.

Loss of information due to data compression may be another explanation for the PDS behavior. Heuristics are condensed form of data cases (or training examples). Thus, bias from either representation or learning algorithm may degrade the quality of heuristics, which, in turn, affect the PDS

performance. Nonetheless, we can assert that PDS benefits from real time execution as the set of training examples becomes richer and more comprehensive because the inductive learning process employs a feedback mechanism to update the learned heuristics. Therefore, in a real system, the relative performance of the pattern directed scheduling approach should improve continually.

5. Dynamic FFS Scheduling

Flexible Flow Systems have become an increasingly important manufacturing technology (Akella, Choong, and Gershwin 1984; Ball and Magazine 1988; Wittrock 1988). The PDS approach proposed in this paper can be extended to scheduling a flexible flow line, an example of which would be a production process utilizing Surface Mount Technology (SMT) for making printed circuit boards (PCBs). The types of scheduling decisions that need to be taken into account while studying the dynamics of a flexible flow system include *dispatching at machines* and *part release*. Dispatching at machines would be similar to those used for FMS scheduling, except that it would be at a local (machine) level. Part release would have to be implemented similar to dispatching, but involving appropriate additional variables and constraints involving these variables.

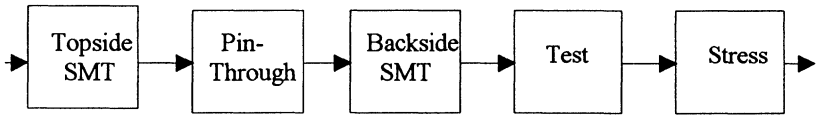
We use data from a major real-world computer manufacturer to model a printed-circuit board assembly facility. The system being modeled is very complex, and every detail from the real-world system is not modeled in this study. We are interested in modeling the system as compact as possible while still maintaining system characteristics that are close to the real system itself. As a first step, we consider only those machines that are prone to bottlenecks in the system at any given moment. Since our primary performance objective in this study, with regard to the system being modeled, is to minimize the mean flow time required of a product that is processed in this system, we schedule only the bottleneck-prone machines and let these schedules take care of the rest of the machines. The rest of the machines do not influence the performance of the system with respect to scheduling, significantly.

The PCB assembly process consists of five stages: Topside SMT, Pin-through Placement, Backside SMT, Testing and Stress. A schematic of this flow line is given in Figure 5-1. We consider the PCB assembly process one step lower, at the machine bank level, specifically those machine banks that are prone to bottlenecks. Although most of the machines in the modeled system require periodic maintenance, we do not model costs or time involved with these in our system. However, we do model the machine failures in our system. The facility being modeled processes two part-families with seven different part-types in each of the part-families, thus adding up to fourteen different part-types that are processed in this system. The different machines that are in this new set being considered have different set-up costs involved depending on the part being

processed and the part waiting to be processed next. The set-up times for various machines arise due to various operations that need to be performed on the machines.

In the course of developing the PDS system, insights into the structure and behavior of flexible flow lines, in the context of PCB fabrication, is obtained. This is particularly true during system design. Increased understanding is gained through abstraction of essential features in the design process to aid in the decisions involved in the system. We briefly discuss the problem specifications that should be taken into consideration while considering a flexible flow system.

Figure 5-1 Schematic of Flow Line for PCB Assembly



5.1 Problem Specification

Scheduling Criteria

Because capital costs are high and variable processing costs are relatively low, high utilization of SMT equipment is a generally accepted goal in the semiconductor industry. The objective of the flexible flow system in this context would be to 1) maximize throughput (or minimize make span), 2)minimize work-in-progress (WIP) and 3) minimize mean flow time (MFT). In this study, we consider minimizing the MFT as the performance criterion since the products are very sensitive to the time spent being processed in the flow shop.

Set of Scheduling Rules

Two different sets of scheduling rules need to be used for FFS scheduling - one for part release, and the other for dispatching. For dispatching, rules such as SIPT can be used similar to those used in FMS scheduling. Rules for part release could include selecting the part that minimizes $max_j \{|x_j(t)|\}$, where $x_j(t)$ is the difference between the total number of parts of type j produced until time t and the total number of parts required, selecting the part that minimizes machine load imbalance, or based on anticipated work in next queue (AWINQ). Whereas scheduling rules for part release are based on a global (system) level,

the rules for machine dispatching operate at a local (machine) level. In this study, we present results from incorporating the part release rules.

Part Release Heuristics

In our study, the part release decision is made whenever the first machine becomes available. The following part release heuristics are considered:
1) *Part Type Priority (PRI) method*: This rule gives a higher priority to the part that belongs to the same family as the part that was released in the immediate past. Ties between parts of the same family are broken on first-come-first-serve (FCFS) basis. This rule aims at minimizing the total number of setups required at the various machines.
2) *Earliest Finish Time at the Bottleneck (EFT-B) method*: This rule gives a higher priority to the part that will complete the earliest at the bottleneck machine. The objective of this rule is to reduce the possibility of starving the bottleneck machine.
3) *Adaptation I of Ow and Smith's (OS-I) method*: This rule adapts the focused scheduling approach proposed in Ow (1985) and Ow and Smith (1988) for the purpose of the SMT system studied here. Suppose that the part release decision is to be made at time t. Following this rule, we next compute the priority $P_i(t)$ of each part i awaiting part release according to the following expression:

$$P_i(t) = \frac{1}{(p_{im} + S_i^+)} exp\left(-\frac{(q - f_i)^+ + h(S_i^+ - S)}{k(\bar{p} + S)}\right)$$

where, m is the bottleneck machine, p_{im} is the processing time of part i at m; t_m is the earliest time at which the next job can be started at m; $S_{i+} = \max(0, t_m + p_{im} - t)$; q is the sum of the average overall processing times (on all the machines) across jobs belonging to different part-types in the system; f_i is the completion time at the last machine. S is the mean forced idle time on m, and p is the mean processing time on m. Ow and Smith [1988] suggest that the performance of their rule is sensitive to the choice of specific values used for parameters h and k. In this study, we use h=20, and k=3 in view of the superior performance obtained under these values in Ow and Smith [1988]. OS-I attempts to combine the benefits of both EFT-B and the well-known shortest processing time (SPT) rule in that it gives a higher priority to the part that has a small overall processing time and that leads to less idle time at the bottleneck machine.
4) *Adaptation II of Ow and Smith's (OS-II) method*: This rule is similar to OS-I above, except that the part priorities are computed as

$$P_i(t) = \frac{1}{(p_{im} + S_i^+)} exp\left(-\frac{(q - f_i - r_i)^+ + h(S_i^+ - S)}{k(\bar{p} + S)}\right)$$

Here, r_i is the total processing time remaining after the bottleneck machine. OS-II differs from OS-I in that it ignores the processing that needs to be done on any part after the bottleneck machine.

In addition to the heuristics listed above, several other heuristics were considered, but these were found to be less effective. These include FCFS (First-Come First-Served), RAN (released in random order), MBPR (part type that is most behind production requirement first - given by the ratio of the number of parts of type i already produced to the total requirement of part-type i), EST-B (release part-types that have the earliest start time at the bottleneck machine first).

Control Attributes

The control attributes for part release and machine dispatching, although mostly the same, differ in that it is at a global level for the former and at a local level for the latter. The control attributes for part release could include total WIP, relative workloads, overall system utilization, contention factor, among others, and those for machine dispatching could include local buffer space, machine utilization, and relative machine load. These attributes that are used for FFS, although the same as those used for FMS scheduling, differ in that both a global average involving all the machines and local values involving those of specific machines are used for FFS. The control attributes selected for this study are: 1) overall system utilization (SU), 2) relative machine workload (NSDRL), 3) contention factor (CFACT), 4) buffer size of each machine (TBF), and 5) bottleneck machine workload (BOTTLE).

These parameters were selected because of their likely impact on MFT following basic queuing theory, as well as other experimental studies conducted previously (see, for example, Baker, 1984; Conway, Maxwell and Miller, 1967).

5.2 Implementation of PDS for FFS

SLAM was used as the modeling environment in this study to simulate the flexible flow system (PCB assembly facility.) During our preliminary analysis of the heuristics that were to be used in the final PDS system, the heuristics that were mentioned in the previous section were narrowed down based on their performances (the mean flow times of the FFS corresponding to the different heuristics are the performance measures in this study, with the heuristic inducing the least mean flow time in the FFS is considered to be the best heuristic for that scenario.) Initially, the part-release heuristic that lead to the best FFS performance (of the four different part-release heuristics), along with their respective parameter (feature) values were collected to be used as the training set of examples. These training set of examples are used as input to the Inductive learning module to generate the {part-release heuristic rules} to be used in the PDS system. Such training examples were used for generating part-release rules.

For considering the part release decision, the FCFS heuristic is used for

dispatching at each machine because it best preserves the order enforced at the time of part release. The tuple comprising a given pattern and the part release heuristic found to be the best for that pattern is recorded for each of the patterns that are generated at this stage to reflect various combinations of the system parameters. These tuples were next input into the inductive learning module to generate the decision tree used by PDS for the part release decision. The decision tree thus generated shown in Figure 5-2 and the heuristic-rules derived from Figure 5-2 are given in Figure 5-3. Note that this tree ignores OS-II even though it is the best part release heuristic for a few patterns. This is a result of tree pruning in order to improve its parsimony. Bottleneck machine workload (BOTTLE) is the variable with the most information content for splitting in this decision tree.

The results using this PDS are encouraging. Initially, the performance of PDS was at least as good as those of the best scheduling rule in 60 percent of the cases. The best scheduling rule is defined as the one which corresponds to the minimum MFT value amongst all the scheduling rules, when used individually. This is encouraging since not all heuristics perform at their best under various circumstances. Also, whenever PDS performed worse than the best scheduling rule, the difference was minimal. The performance of this PDS can be improved further through knowledge refinement process discussed in the next section.

Figure 5-2 Decision Tree for Part Release

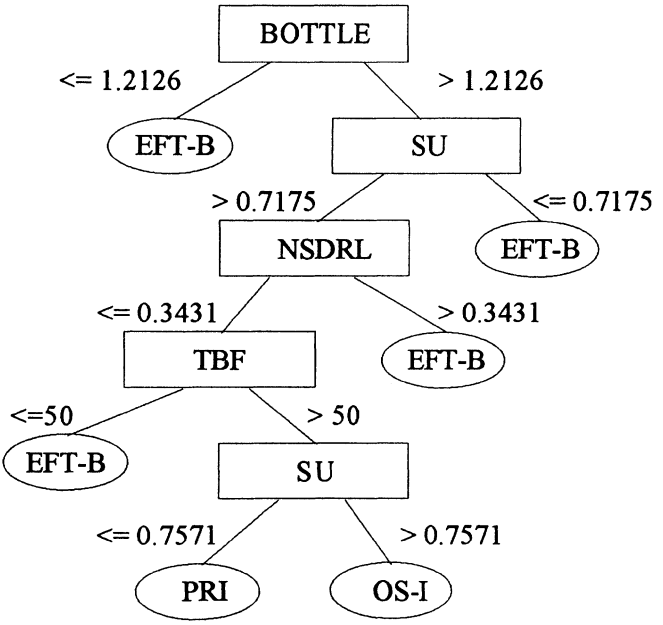


Figure 5-3 Heuristic Rules Derived from Figure 5-2

IF (BOTTLE <= 1.2126) THEN EFT-B.
IF (BOTTLE > 1.2126) AND (SU <= 0.7175) THEN EFT-B.
IF (BOTTLE > 1.2126) AND (SU > 0.7175) AND (NSDRL > 0.3431) THEN EFT-B.
IF (BOTTLE > 1.2126) AND (SU > 0.7175) AND (NSDRL <= 0.3431) AND (TBF <=50) THEN EFT-B.
IF (BOTTLE > 1.2126) AND (SU > 0.7175) AND (NSDRL <= 0.3431) AND (TBF >50) AND (SU > 0.7571) THEN OS-I.
IF (BOTTLE > 1.2126) AND (SU > 0.7175) AND (NSDRL <= 0.3431) AND (TBF >50) AND (SU <= 0.7571) THEN PRI.

6. Adaptation by Learning and Knowledge Refinement

An intelligent system needs to be adaptive to its current environment for it to perform effectively even as the environment evolves. By adapting to its current environment, the system can react to the problem solving situation by utilizing appropriate models and strategies (scheduling rules in FMS and FFS scheduling system), rather than ones that are out-dated or otherwise inappropriate to the current situation.

Machine learning is used in this study as a means by which the system adapts itself to the environment. Machine learning is used to continually update and refine the knowledge-base. The critic module monitors the quality of the schedules generated at the pattern-directed scheduling phase by comparing its performance with those obtained by repeatedly applying each scheduling rule individually under a variety of scenarios. If a learned heuristic rule is proved to be causing undesirable scheduling performance, the rule-refinement module is initiated. For example, higher mean tardiness values under PDS indicate deficiencies that need correction. These deficiencies could be cased by not considering a large enough set of training examples. As noted earlier, it is difficult for the set of training examples to be comprehensive in view of vastness of the system attribute space. Consequently, the heuristic selection rules imbedded in the decision tree are over generalized to some extent. If this results in performance degradation, then the heuristics need to be refined.

The rule refinement procedure identifies all such over generalized conditions, and augments the decision tree by generating additional rules appropriately (specialization). The metric used for rule refinement is prediction accuracy α that measures the proportion of testing observations in which the scheduling rule selected by the decision heuristics turns out to be the one that performs the best among all scheduling rules when implemented individually. On a random sample of test observations, if α is found to be less than *thv*, the pre specified target prediction accuracy level, then the learned heuristics in the decision tree are refined following a three step process. First, deficient heuristic

rules in the decision tree are identified; next, additional training examples are generated in order to specifically address preconditions manifested in the deficient rules. In the final step, the inductive learning algorithm is initiated to update the tree on the basis of the additional information provided by these examples. The process of generating the testing instances, evaluating α on these instances, and refining the tree is carried out repeatedly until the desired prediction accuracy is achieved. For the formal description of this refinement process, refer to Park, Raman, and Shaw 1993 and Park, Kim, and Shaw, 1993.

7. Concluding Remarks

We have developed an intelligent scheduling decision support system with learning capabilities using inductive learning, an artificial intelligence methodology, for decision support. The proposed system learns from past experience through experimentation in the domain of interest. The learning process also endows the system with automated knowledge acquisition capability, thus avoiding the hard task of manually acquiring problem-related knowledge. Knowledge thus acquired is also refined through feed-back from a critic module utilizing pre-specified performance criteria.

This paper has established the following results which we think are significant from a intelligent scheduling perspective:
1) It is desirable to apply appropriate scheduling rules adaptively as per the needs of the manufacturing system, 2) the heuristics for selecting best scheduling rule can be generated by the inductive learning process, and 3) a new methodology for improving the performance of the system by integrating simulation and inductive learning, resulting in an adaptive intelligent scheduling system which learns and evolves over time through experimentation.

We used the problem of scheduling in a FMS and a FFS to illustrate the proposed PDS system. By utilizing the PDS recognized as an intelligent scheduling system, scheduling in an FMS and an FFS environment could be done adaptively as per the dynamics of the system.

References

Akella, R. Choong, Y., and Gershwin, S.B., "Performance of Hierarchical Production Scheduling Policy," *IEEE Transactions on Components, Hybrids, and Manufacturing Technology*, vol. CHMT-7, no. 3 pp. 225-240, Sep. 1984.

Baker, K.R., "Sequencing Rules and Due-Date Assignments in a Job Shop," *Management Science*, Vol. 30, No. 9, pp. 1093-1104, Sep. 1984.

Ball, M. O., and Magazine, M.J., "Sequencing of Insertions in Printed Circuit Board Assembly," *Operations Research*, 36, pp. 192-201, 1988.

Breiman, L., Friedman, J., Olshen, R., and Stone, C., *Classification and Regression Trees*, Belmont, CA, Wadsworth, 1984.

Buzacott, J., "Optimal Operating Rules for Automated Manufacturing Systems", *IEEE Transactions on Automatic Control*, Vol. AC-27, 80-86, 1982.

Chaturvedi, A.R., Hutchinson, G.K., and Nazareth, D.L., "Supporting Complex Real-time Decision Making through Machine Learning," *Decision Support Systems*, vol. 1, no. 2, pp. 213-234, 1993.

Conway, R.W., Maxwell, W.L., and, Miller, L.W. *Theory of Scheduling*, Addison-Wesley, Reading MA, 1967.

Gunasekaran, A., Martikainen, T., and Yil-Olli, P., "Flexible Manufacturing Systems: An Investigation for Research and Applications", *European Journal of Operational Research*, vol. 66, no.1, pp. 1-26, 1993.

Holsapple, C.W., Pakath, R., Jacob, V.S., and Zaveri, J.S., "Learning by Problem Processors: Adaptive Decision Support Systems," *Decision Support Systems*, vol. 1, no. 2, pp. 85-108, 1993.

Hong, J., Mozetic, I., and Michalski, R.S., "AQ15: Incremental Learning of Attribute-Based Descriptions from Examples," *Technical Report No. UIUCDCS-F-86-949*, Dept. of Computer Science, University of Illinois at Urbana-Champaign, July, 1986.

Mitchell, T.M., Keller, R.M., and Kedar-Cabelli, S.T., "Explanation Based Learning: A Unifying View," *Machine Learning*, vol. 1, pp. 47-80, 1986.

Ow, P.S., "Focused Scheduling in Proportionate Flowshop," Management Science, vol. 31 no. 7, pp. 852-869, 1985.

Ow, P.S., and Smith, S.F., "Viewing Scheduling as an Opportunistic Problem-solving Process," Annals of Operations Research, vol. 12, pp. 85-108, 1988.

Park, S.C., "Applying Machine Learning to the Design of Decision Support System for Intelligent Manufacturing," *Ph.D. Dissertation, University of Illinois - Urbana & Champaign*, 1991.

Park, S.C., Kim, J., and Shaw, M.J., "A Comparative Study on Rule Refinement", *Wisconsin Working Paper 2-94-6* University of Wisconsin-Madison, School of Business, 1993.

Park, S.C., Raman, N., and Shaw, M.J., "Adaptive Scheduling in Dynamic Flexible Manufacturing Systems," *Wisconsin Working Paper 12-93-14*,

University of Wisconsin-Madison, School of Business, 1993.

Park, S.C., Piramuthu, S., Raman, N., and Shaw, M.J., "Integrating Inductive Learning and Simulation in Rule-based Scheduling," in Gottlob, G., and Nejdl, W. (eds.), *Lecture Notes in Artificial Intelligence #462: Expert Systems in Engineering - Principles and Applications*, pp. 152-167, 1990.

Park, S.C.., Raman, N., and Shaw, M.J., "Heuristic Learning in Pattern-Directed Scheduling," in *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, Elsevier Science, Amsterdam, Netherlands, 1989.

Pierreval, H., "Rule-based Simulation Metamodels", *European Journal of Operational Research*, vol. 61, no.1-2, pp. 6-17, 1992.

Piramuthu, S., Park, S.C., Raman, N., and Shaw, M.J., "Integration of Simulation Modeling and Inductive Learning in an Adaptive Decision Support System, *Decision Support Systems*, vol. 9, pp. 127-142, 1993.

Quinlan, J.R., "Decision trees and Multi-Valued Attributes," *Machine Learning*, vol.1, pp. 305-318, 1988

Quinlan, J.R., "Induction of decision Trees," *Machine Learning*, vol. 1, pp. 81-106, 1986.

Rendell, L., "Induction, of and by Probability," in Kanal and Lemmar (Eds), *Uncertainty in AI*, pp. 429-443, North Holland, 1986.

Shaw, M. J., "A Pattern-Directed Approach to Flexible Manufacturing: A Framework for Intelligent Scheduling, Learning, and Control," *International Journal of Flexible Manufacturing Systems*, Vol. 2, pp. 121-144, 1989.

Shaw, M.J., "Knowledge-Based Scheduling in Flexible Manufacturing Systems: An Integration of Pattern-Directed Inference and Heuristic Search," *International Journal of Production Research*, Vol. 15, No. 5, pp. 353-376, 1988.

Shaw, M. J., S. C. Park, and N. Raman, "Intelligent Scheduling with Machine Learning Capabilities: The Induction of Scheduling Knowledge," *IIE Transactions*, vol.24, no.2, pp. 156-168, 1992.

Wittrock, R.J., "An Adaptive Scheduling Algorithm for Flexible Flow Lines", *Operations Research*, Vol. 36, No. 3, pp. 445-453, 1988.

Solving Large Integer Programs Arising from Air Traffic Flow Problems

RUSTY BURLINGAME[1], E. ANDREW BOYD[2], and KENNETH S. LINDSAY[3]

[1] *Department of Industrial Engineering, Texas A&M University, 238 Zachry Building, College Station, Texas 77843-3131,* rusty@marvin.tamu.edu

[2] *Department of Industrial Engineering, Texas A&M University, 238 Zachry Building, College Station, Texas 77843-3131,* boyd@marvin.tamu.edu

[3] *The MITRE Corporation, Center for Advanced Aviation System Development, 7525 Colshire Drive, McLean, Virginia 22102-3481,* klindsay@mitre.org

Abstract

Recent integer programming techniques are used to solve problems arising in the control of national air traffic. Two large problems based on real data and developed by the MITRE Corporation for the Federal Aviation Administration are solved to provable optimality. The work was motivated by the inability to solve these problems using commercial integer programming codes. The results demonstrate that, through effective use of recent integer programming techniques, air traffic control problems can be solved that could not be solved without these techniques.

key words: integer programming and flight scheduling

1. Introduction

From take-off to landing, every commercial aircraft flying between points in the continental United States is under the control of the U. S. Air Traffic Control system. According to Zenios (1990), on an average day in 1984 there were some 17,000 such flights, with expected volume continuing to rise at a rapid pace well into the next century. The job of controlling this system is extremely complex with conflicting objectives. Not only are air traffic controllers expected to maintain adequate separation between aircraft to ensure safety, but also to allow aircraft to reach their intended destinations in the most direct fashion with the least possible delay. Further, the problem is constantly changing as airport capacities are adjusted for inclement weather. Given the size of the system, the cost of excess fuel consumption from flights delayed or rerouted can potentially run into millions of dollars a day, the cost of which is a burden upon airlines and ticket buyers alike. According to Vranos, Bertsimas, and Odoni (1992), in 1986 total delay costs in U. S. air space approached the $2 billion mark while total profits for U. S. carriers during the same period were roughly $800 million. It can only be expected that such costs will continue to rise while increasing air traffic volume places an ever greater strain on the existing air traffic facilities.

Many mathematical programming models have been proposed for controlling air traffic. While they differ in their formulations and to some extent in the goals they seek to achieve, common to almost all of the models is that they are very large scale integer or mixed integer programs. As a general rule, the larger the problems that can be solved, the more useful the results of the model. Ideally, it would be desirable to solve problems with tens or hundreds of millions of variables and hundreds of thousands of constraints. Of course, this is a potentially herculean task as there exist integer programs with as few as 100 variables that have never been solved to optimality.

The purpose of the present paper is to demonstrate how recent integer programming techniques can be used to solve large air traffic control problems. Two large problems based on real data and developed by the MITRE Corporation for the Federal Aviation Administration are solved to provable optimality. The work was motivated by the inability to solve these problems using commercial integer programming codes. The results demonstrate that, through effective use of recent integer programming techniques, air traffic control problems can be solved that could not be solved without these techniques.

As early as 1957, Ferguson and Dantzig (1957) attempted to model and solve air traffic control problems, but real activity in the area did not begin in earnest until the 1980's. In 1983, Booth and Harvey (1983) proposed a large mixed-integer linear programming formulation for minimizing air traffic delay and maintaining safety standards but attempted no com-

putational experiments. Mulvey and Zenios (1987) proposed a nonlinear generalized network model for minimizing air traffic delay and maintaining safety standards, and in related articles (1985), one with coauthors Ahlfeld and Dembo (1985), they describe a powerful code for nonlinear generalized network problems and apply this algorithm to solve some instances of their model based on real problem data. The largest problem they solved consisted of 1295 constraints and 2873 variables and was solved in a few seconds on a CRAY supercomputer. The work is impressive, clearly indicating the speed of the nonlinear generalized network solver they developed. However, the problem remains relatively small, and the issue of integrality of the variables is apparently achieved through an implicit rounding procedure. Vranas, Bertsimas, and Odoni (1992) perform computational experiments on some larger problems with explicit consideration given to integrality of the variables. While their study is extensive and thoughtful, the problems they solve are based on randomly generated data.

The computational work presented in this present paper differs from these previous studies in a number of ways. Unlike the Zenios and Mulvey study, integrality of the variables is treated explicitly, and the models that are solved are larger. While the models are roughly similar in size to those solved by Vranas, Bertsimas, and Odoni, they have the important difference of being based on real problem data.

The paper is organized as follows. In the following section, the underlying problem and formal model are introduced and computational results are presented to demonstrate the difficulty of solving this model without further refinement. In the third section, the preprocessing techniques used to reduce the problem size are discussed. We then describe the branch-and-bound procedure used in conjunction with the preprocessing techniques, and computational results are presented to demonstrate the effectiveness of the proposed methodologies. The paper concludes with a discussion of further research issues.

2. The Problem

While the focus of the present paper is on computational techniques rather than the model these techniques were used to solve, it is useful to understand the underlying problem and formal model before discussing computation. Greater detail on the model itself can be found in a MITRE Letter by Lindsay (1992).

In the absence of congestion, every commercial airline flight would be allowed to depart at its scheduled time and to proceed to its destination and land without any delays. Clearly, airports provide an obvious source of congestion, especially under the present hub-and-spoke system which causes great congestion at hub airports during peak traffic periods. Less obvious but just as important is congestion experienced in the air. National airspace is divided into approximately 500 sectors, with an air traffic con-

troller in charge of directing flights safely through each sector. One method of minimizing risk is simply to limit the number of flights in any sector at any given period of time.

The Time Assignment Model seeks to assign the times at which a flight is to be at given fixes in its flight path. A fix is a point in space, that is, an arrival airport, a departure airport, or an arbitrary point in the aircraft's flight path. The time assignment is achieved by first evenly dividing the time horizon of interest, T, into j equal intervals T_j. The flight path of each of the N flights F_i is specified by an ordered set of fixes f_i, beginning with the point of departure and ending with the destination. For example, flight F_3 could take off at time T_{11} from fix $k_6 \in f_3$ and be at fix $k_7 \in f_3$ by time T_{14}. A 0/1 decision variable is then introduced for each combination of flight, time period, and fix included in the space-time trajectory for that flight. Flights must traverse each fix in their specified flight path in a particular order, not departing the initial fix prior to a set departure time. Also, flights cannot travel from one fix to the next in less than a specified time (the minimum time required to travel between the two fixes). Flights may depart late from the initial fix (ground delay) at a given penalty, and may travel at a reduced speed between fixes (air delay), also at a penalty. Generally, air delay penalties are higher since they involve higher fuel costs.

Formally, the model can be expressed as follows:

$$\min \sum_i \sum_{k_n \in f_i} C_{ik_n} \left(\sum_j T_j \left(x_{ijk_n} - x_{ijk_{n-1}} \right) - t_{ik_{n-1}k_n} \right) \tag{1}$$

for each pair of consecutive fixes $k_{n-1}, k_n \in f_i$.

Subject to the following constraints:

$$\sum_i x_{ijk} \leq c_{jk} \quad \forall\, j, \forall\, k \tag{2}$$

$$\sum_j T_j \left(x_{ijk_n} - x_{ijk_{n-1}} \right) \geq t_{ik_{n-1}k_n} \quad \forall\, i, \forall\, k \in f_i \tag{3}$$

$$\sum_j x_{ijk} = 1 \quad \forall\, i, \quad \forall\, k \in f_i \tag{4}$$

$$x_{ijk} = 0 \text{ or } 1$$

Problem constants:

f_i : ordered set of fixes of flight i

C_{ik} : cost per unit time of delaying flight i at fix k

c_{jk} : capacity at fix k during time period j

$t_{ik_{n-1}k_n}$: minimum time required for flight i to fly from fix $k_{n-1} \in f_i$ to the next fix $k_n \in f_i$

T_j : point time surrogate for time interval T_j

	TFM2	TFM3
variables	10274	27300
constraints	2054	3560
constraint nonzeros	39704	100860
objective nonzeros	9324	16680
RHS nonzeros	2054	3505
greater than constraints	383	910
equality constraints	383	910
less than constraints	1288	1740
flights	113	278
time periods	28	30
unique fixes	44	48
LP relaxation solution	114815.96...	33990.0
integer optimal solution	115155.0	33990.0
fractional variables in relaxation solution	791	1547

Table 1: Statistics for Integer Programs TFM2 and TFM3

Decision variables:

$$x_{ijk} = \begin{cases} 1 & \text{if flight } i \text{ is to be at fix } k \text{ during time period } T_j \\ 0 & \text{otherwise} \end{cases}$$

The constraints (2) represent capacity restrictions at a given fix and given point in time, while the constraints (3) represent the minimum flight time restrictions between two fixes. The constraints (4) simply require that every flight must traverse each specified fix at some time.

Two test problems named TFM2 and TFM3 based on real flight data were generated by MITRE. Table 1 contains some relevant statistics on the two problems. TFM3 is not only the larger problem, but is also more congested and consequently more difficult to solve. The greater level of congestion was verified by the optimal solutions to the problems, as flights were delayed at most two time periods for TFM2 and up to four time periods for TFM3. TFM3 also proved to be extremely dual degenerate at the optimal solution, adding to the difficulty of solving this problem since the reduced costs provided little if any useful information.

All computational work was performed on a dual processor SUN SPARC-server 670 with 64 megabytes of main memory. Attempts were made to solve both of the problems using the integer programming capabilities of the commercial optimization package CPlex. Due to the large number of fractional variables in the optimal solutions to both problems, it was expected that the CPlex branch-and-bound algorithm would have difficulty solving them directly and this was indeed the case. For TFM3, CPlex ran

for 6 hours before quitting after having generated 20,000 branch-and-bound nodes. Because of the congestion implicit in the model, so much branching occurred under the default branching strategy that the algorithm never ventured far enough down the branch-and-bound tree to generate a *feasible* solution. A depth-first strategy ran for over 2 hours before running out of memory. Over 10,000 branches were searched, and again no feasible integer solution was found, apparently because a wrong branch was taken early in the branching process. Since the problem is so degenerate, it is nearly impossible to tell when a wrong branch is chosen, or to identify the incorrect branch after it becomes obvious that a wrong choice was made. The branch-and-bound algorithm slowly backed up, filling the entire tree below the incorrect branch. Even when there are only a few delays, as in the TFM2 problem, branch-and-bound cannot solve the problem in 20,000 nodes, for the same reasons.

Alternatively, an attempt was made to solve both TFM2 and TFM3 using the general purpose preprocessing subroutine supplied as part of IBM's Optimization Subroutine Library (OSL). Here, some success was achieved. Although a publically released version of OSL was not able to solve the smaller problem TFM2 correctly, with the help of IBM's staff an unreleased version of the subroutine was able to solve TFM2 in a few minutes on an IBM RISC Station after some initial bugs were ironed out. On the larger more difficult problem TFM3, however, the routine ran for a number of hours before exhausting available memory and terminating.

3. Preprocessing

Preprocessing techniques were first popularized by Crowder, Johnson, and Padberg (1983) in their award winning work. Hoffman and Padberg (1991) later extended this work, focusing solely on the issue of preprocessing. Generally speaking, preprocessing techniques consist of a variety of simple, fast methods of reducing the size of an integer program by fixing variables, removing constraints, and modifying constraint coefficients to strengthen the integer programming formulation. While the techniques are mathematically quite rudimentary, they can prove to be remarkably successful, especially when used in conjunction with cutting planes and well-designed branch-and-bound strategies. It is important to emphasize that while preprocessing techniques are all quite simple and of limited use in and of themselves, when used in conjunction they can have a cascading effect that substantially simplifies a problem. What is important is finding a collection of such techniques that reduce problem size the most effectively. This section discusses the preprocessing techniques that proved effective in solving time assignment problems.

Both general and problem specific preprocessing techniques were applied to the time assignment problems. With the variables restricted to take on the values 0 or 1, it was easily determined if a less than or equal to con-

straint was redundant by checking if the sum of the positive coefficients was less than or equal to the right-hand-side. Such constraints were removed from the problem, reducing its size. Greater than or equal to constraints were removed similarly. Further, whenever a constraint or a collection of constraints were removed, variables were potentially removed as well, or could be fixed at the obvious value if they only appeared in the objective function.

Other methods of variable fixing also proved effective. Variables can be fixed if a constraint can only be satisfied by one configuration of variables. For example, given an equality or less than or equal to constraint with all positive variable coefficients and a right-hand-side of 0, it can be concluded that all of the variables involved in the constraint must be 0. This type of situation arose, for example, when one of the variables in a type 4 constraint was fixed at 1.

A general technique that was not employed but is often quite useful is a method of fixing variables when the absolute value of their reduced costs is greater than the gap between the value of the linear programming relaxation and the value of a known feasible integer solution (see Crowder, Johnson and Padberg 1983 or Hoffman and Padberg 1991 for more details). The major practical difficulty with applying this technique is that it requires knowledge of a good feasible integer solution prior to actually solving the problem. While this technique was experimented with after the optimal integer solution was known and proved to be of some value in solving TFM2, it was not useful in solving TFM3, and the branch-and-bound method described in the following section proved equally effective without reduced cost variable fixing. The primary difficulty with successfully employing this technique was that a large proportion of the non-basic variables had reduced costs of zero. This was not surprising given that the very nature of the model is such that there are likely to be many alternative optimal solutions. It is expected that larger problems will have an even greater multiplicity of optimal solutions, and therefore reduced cost variable fixing will not help.

In addition to the general techniques outlined above, a technique specific to the type 3 constraints was also used with great success. Recall that the type 3 constraints focus on two consecutive fixes for a particular flight, and the possible times that the plane may be at each fix. These constraints are of the form:

$$\sum_j T_j \left(x_{ijk_n} - x_{ijk_{n-1}} \right) \geq t_{ik_{n-1}k_n} \quad \forall\, i,\ \forall\, k \in f_i$$

We can look at each individual type 3 constraint (that is, choose i_m, k_n, and k_{n-1}) and note that, because of the type 4 constraints,

	cons	vars	con nonzeros	obj nonzeros
TFM2 before	2054	10724	39704	9324
TFM2 after	1132	4946	15778	4323
% reduction	44.9	43.9	60.3	53.7
TFM3 before	3560	27300	100860	16680
TFM3 after	1965	11589	35741	6994
% reduction	44.9	57.6	64.6	58.1

TFM2 Preprocessing Time: 47 seconds
TFM3 Preprocessing Time: 1775 seconds

Table 2: Effect of Initial Preprocessing

$$\exists\, p \text{ such that } x_{i_m j_p k_n} = 1$$

and

$$\exists\, q \text{ such that } x_{i_m j_q k_{n-1}} = 1.$$

In other words, the plane must pass through each fix in its flight path some time. In the case of a plane flying from fix k_p to fix k_q with flight time t, the *sequence* of type 4 constraints for a given flight enforce the following two rules.

- Since there is a latest possible time that the plane may reach fix k_q, the plane must leave fix k_p at least t time units before then. Variables which placed the plane at fix k_p after such time were removed.
- Since there is an earliest time that the plane may leave fix k_p, the plane must arrive at fix k_q at least t time units after that. Variables which placed the plane at fix k_q before such time were removed.

Stepping through the type 3 constraints in this manner had a cascading effect on the problem, and several passes were needed to remove all possible variables. In addition, when a type 3 constraint is found to have only two remaining variables, it follows that both must take on a value of 1, fixing two variables and making the constraint redundant. This observation was also used to fix variables and remove constraints.

The techniques just described were employed until no further variables could be fixed or constraints removed. Statistics on the reduced problems are shown in Table 2. It is important to note that in the present work no effort was made to design the preprocessing algorithms for speed as emphasis was instead placed on finding the right collection of preprocessing routines.

4. Branch-and-Bound Strategies

Following preprocessing, the linear programming/integer programming gap was reduced to 0 for both TFM2 and TFM3. Unfortunately, this was not known during algorithm development since there still remained a large number of fractional variables in the optimal solution. Even with this knowledge, however, it remains very difficult to determine an optimal integer solution. The default branch-and-bound strategies of CPlex still fail to generate feasible solutions for either of the problems.

Efforts were therefore made to develop a branch-and-bound strategy that took advantage of the special problem structure of TFM2 and TFM3. In order to minimize the delays, and therefore the costs associated with those delays, it seems natural that each plane should be scheduled to be at each fix as early as possible. This time can be inferred from the type 4 constraints which list all possible times that a certain plane may be at a certain fix. The obvious branching strategy would be to branch on the variables with least delay first. Unfortunately, the type 2 congestion constraints rendered this method ineffective. Branch-and-bound continued to make incorrect branches and failed to recover from such branches.

Interestingly, the strategy that proved most effective was exactly the opposite: *to branch on variables with the longest delay first.* For example, some variables in the problem TFM3 allow a plane to delay at a fix for up to 26 time periods. It is highly unlikely that an optimal solution to a real problem would require such a long delay (although it is not at all clear how many time periods should appear in the original model — too few would yield an infeasible problem). Thus, while it is very hard to choose which variables are definitely part of an optimal solution, it is much easier to choose variables which are not. The implied branching strategy is to perform a 0 branch on variables corresponding to a long delay. The advantage to this approach is that once these variables have been branched upon, preprocessing techniques can once again be employed to the reduced problem. While the results of the preprocessing are not valid for the original problem, they are valid for the given node of the branch-and-bound tree. In a worst case scenario this "branch-and-preprocess" approach could be used throughout the tree, but this did not prove necessary.

The aforementioned strategy was easily implemented using the CPlex callable library. It was necessary to create a priority list in order to control the order in which CPlex would branch upon variables. Since it was desired to branch first on variables that corresponded to a large delay, these variables were assigned a high priority. A highly successful priority list was created by assigning each variable a priority equal to the amount of delay corresponding to that variable. In addition to using this priority list, it was also found necessary to override the default settings guiding the branch-and-bound strategy, which had not proved successful in early solution attempts. CPlex was set to branch by fixing variables to 0 rather than

	cons	vars	con nonzeros	obj nonzeros
TFM2 before	1132	4946	15778	4323
TFM2 after	1132	1532	4791	1332
% reduction	0.0	69.0	69.6	69.2
TFM3 before	1965	11589	35741	6994
TFM3 after	1965	3579	10728	2183
% reduction	0.0	69.1	70.0	68.8

TFM2 0 Branching Time: 26 seconds
TFM3 0 Branching Time: 237 seconds

Table 3: Effect of 0 Branching

	cons	vars	con nonzeros	obj nonzeros
TFM2 before	1132	1532	4791	1332
TFM2 after	752	1532	4310	1332
% reduction	33.6	0.0	10.0	0.0
TFM3 before	1965	3579	10728	2183
TFM3 before	1700	3547	9621	2151
% reduction	13.5	0.9	10.3	1.5

TFM2 Second Preprocessing Time: 7 seconds
TFM3 Second Preprocessing Time: 30 seconds

Table 4: Effect of Second Preprocessing

1 (when there was a choice) and to perform a depth-first search. Taken as a whole, this strategy was remarkably successful, especially in light of our initial attempts at using branch-and-bound.

The use of preprocessing within the branch-and-bound tree proved to be the key to solving TFM2 and TFM3, and was absolutely essential for solving TFM3. Results are shown in Tables 3 and 4. Complete solution times are given in Table 5. This table includes the time for the initial preprocessing, as discussed in the previous section, solution time for the resulting linear program, and branching times as discussed in this section. Preprocessing alone was not adequate to find an optimal solution, but after only 356 nodes for TFM3 and only 23 for TFM2, an integer solution was found with the same value as the linear programming relaxation, and it was thus possible to conclude that the solution was optimal. In the problem TFM2, planes were delayed at most two time periods, but in TFM3 some planes were delayed up to four time periods. These figures represent a

	TFM2	TFM3
Initial Preprocessing	47	1775
Initial Linear Program	23	85
0 Branching Time	26	237
Second Preprocessing	7	30
Branch-and-Bound	11	151
Total (seconds)	114	2278

Table 5: Computation Time Summary

significant level of congestion for the algorithm to have resolved to achieve provable optimality.

5. Conclusions

The purpose of this paper was to demonstrate how recent integer programming techniques could be used to solve large air traffic control problems. The effectiveness of these techniques was demonstrated by applying them to solve two large integer programs which could not be solved with a commercial branch-and-bound algorithm, even when employing an intelligent branching strategy.

While the problems that were solved were large by integer programming standards, it would be useful to solve problems that are 10 to 100 times larger. The present computational results suggest that this may in fact be possible, but to do so will require more efficient implementation of the ideas described in this paper. Also, it is likely that larger problems will give rise to new challenges requiring further preprocessing techniques and alternative branching strategies. The authors hope to address these issues in future research.

Bibliography

Ahfeld, D. P., R. S. Dembo, J. M. Mulvey, and S. A. Zenios, "Nonlinear Programming on Generalized Networks," Technical Report EES-85-7, Department of Civil Engineering, Princeton University, Princeton, New Jersey, 1985.

Booth, G. R. and C. W. Harvey, "Minimizing air traffic delays by mathematical programming," in: *Safety issues in air traffic system planning design*, Princeton University, Princeton, New Jersey, 1983.

Crowder, H., E. L. Johnson, and M. W. Padberg, "Solving large-scale zero-one linear programming problems," *Operations Research*, vol. 31, pp. 803-834, 1983.

Ferguson, A. R. and G. B. Dantzig, "The allocation of aircraft to routes," *Management Science*, vol. 3, pp. 45-73, 1957.

Hoffman, K. L. and M. Padberg, "Improving LP-representation of zero-one linear programs for branch-and-cut," *ORSA Journal on Computing*, Vol. 3, No. 2, pp. 121-134, 1991.

Lindsay, K. S., Transmittal of Revision of National Traffic Flow Management (NTFM) Time Assignment Model, Enclosures 1 and 2, to Professor E. A. Boyd, MITRE Letter F049-L-006, MITRE Corporation, McLean, Virginia, 1992.

Mulvey, J. M. and S. A. Zenios, "Real-time operational planning for the U. S. air traffic system," *Applied Numerical Mathematics*, vol. 3, pp. 427-441, 1987.

Mulvey, J. M. and S. A. Zenios, "Solving large scale generalized networks," *Journal of Information and Optimization Science*, vol. 6, pp.94-111, 1985.

Vranas, P. B., D. J. Bertsimas, and A. R. Odoni, "The multi-airport ground-holding problem in air traffic control," Technical Report OR 263-92, Operations Research Center, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1992.

Zenios, S. A., "Network based models for air-traffic control," *European Journal of Operational Research*, vol. 50, pp. 166-178, 1990.

Zenios, S. A. and J. M. Mulvey, "Nonlinear network programming on vector supercomputers: A study on the CRAY X-MP," Technical Report EES-85-13, Department of Civil Engineering, Princeton University, Princeton, New Jersey, 1985.

INTELLIGENT SCHEDULING SUPPORT FOR THE U.S. COAST GUARD

Ken Darby-Dowman[1], Cormac Lucas[1], Gautam Mitra[1]
Raymond Fink[2], Leonard Kingsley[3], J. Walter Smith[3]

[1] Department of Mathematics and Statistics, Brunel University, Uxbridge, Middlesex, UB8 3PH, U.K.

[2] Idaho National Engineering Laboratory
EG&G Idaho, Inc, P.O. Box 1625, Idaho Falls, ID 83415-2407, USA

[3] United States Coast Guard R & D Center, Avery Point, Groton, Connecticut 06340-6096, USA

Abstract
A hybrid knowledge base and integer programming decision support system for scheduling US Coast Guard cutters is presented. The optimization model is essentially an integer goal programming formulation which determines a day-by-day schedule over a year for each cutter in the fleet. This is achieved by generating a large number of possible schedules for each cutter and optimizing the overall fleet schedule by selecting one schedule for each cutter. The model is robust and is guaranteed to produce schedules even when, as is generally the case, the schedule requirements conflict. The knowledge base is used in the schedule generation and also allows schedule evaluation to take place. The system has been implemented and field trials show that, in addition to operational use, it has an important role in strategic policy evaluations. Design and computational issues are discussed.

key words: Integer programming, Knowledge based systems, Optimization, Scheduling.

1. Background

The United States Coast Guard (USCG) undertakes several important missions in the maritime regions of this country. These responsibilities are classified as:

(i) Search and rescue.

(ii) Law enforcement.

(iii) Response to (marine) environmental incidents.

(iv) Fishery and custom regulation.

(v) Vessel safety.

A number of platforms, that is, vessels (cutters) and aircraft of different types are normally deployed to carry out these missions as appropriate. In recent times requests for these platforms have increased tremendously. For instance, the growing usage of small boats by persons without adequate nautical training has increased the search and rescue missions substantially. Law enforcement requirements have also expanded to combat drug related activities and to check the rivalries of different nations in the use of natural resources at sea. These growing demands call for better usage of already scarce sea and air platforms which are at the disposal of the USCG. Thus, the USCG operational staff are required to make efficient scheduling decisions under severe time, fiscal and legal constraints.

The long-term planning and scheduling of diversified Coast Guard platforms, engaged in a variety of missions, is a complicated process. Thus the Coast Guard has assigned the scheduling offices so platforms having similar operating profiles are grouped together to simplify the processes. Platforms are further separated on the east and west coasts under Commands which are called LANTAREA (Atlantic Area) and PACAREA (Pacific Area). Moreover, cutters and aircraft are scheduled by different offices within the Area commands. Each Area is further subdivided into several districts. The District commanders manage all operational and administrative matters arising in their districts. The Area Commander, however, regulates all operational missions that are "multi-district" in geographical region, scope, or resources.

Periodically, the Area Commander's staff prepares and formally issues operational schedules for the platforms under their regional control. These schedules serve as the operational orders to the Commanding Officers of the platforms, specifying the duties that the platform is required to perform at a given time, duration and location. Moreover, the schedule has a large impact on morale, informing the crews when they can expect to be home with their families, and when and for how long they will be away.

Typical schedules include assigned durations for maintenance, patrols, public affairs events, and training and other exercises. Moreover, specific doctrines impose restrictions on the number of days a particular platform may be away from homeport and the minimum number of days that a platform can be assigned to a particular patrol location. In addition, the intervals between training and maintenance actions are fixed by Coast Guard policy. However, contracting maintenance facilities for engine overhauls and technology upgrades result in uncertainties in the starting and ending dates of a given assignment producing volatile schedules. Further, unscheduled maintenance and emerging mission requirements may cause adjustments to be made to an existing schedule as platforms are reassigned to unscheduled activities. Much greater effects can be caused by such extremes as a military action (that requires Coast Guard assets) or a sudden loss in the budget for operating funds.

Individual platforms, scheduled by separate offices, frequently must work with one another on particular missions. For example, some cutters are equipped with a flight deck for carrying helicopters on patrol. A conflict can occur if the helicopter is unavailable during a cutter/helicopter patrol assignment. Likewise, various training exercises might require platforms which have been scheduled separately by other authorities, that is, U.S. Navy assets. So, even though schedules are developed independently, the appropriate scheduling office must operate within an environment that contains a high degree of cooperation. This poses a burden on managers in the separate offices faced with decision making under constraints of time and conflicting interests.

Taking the above factors into account in the schedule production process, while trying to increase the operational time for each platform, was a difficult and tedious task. In the past, the schedule was produced manually (with pencil and paper). Adjusting the basic input repeatedly, to produce a more

accurate schedule, required many professional staff hours. So, the quality of the schedule, measured as a function of cost and platform utilization, was degraded by this inability to produce new schedules, as the old ones became infeasible, in a timely manner. In addition, examining proposed changes to policy such as increasing operational day limits within a particular patrol location, required generating new schedules to test out the impact of these changing policies.

The scheduling authority's stated goals are to write a "maximum utilization" schedule, provide statistics such as the number of operational days per quarter for a particular platform and offer alternative solutions based upon changes in scheduling doctrine. Historically, the Coast Guard relied on the skill and experience of human schedulers alone to "load" the platforms with assignments. However, as the number of platforms grew, the scheduling process became so complex that human schedulers could not handle the process without being aided by the latest computer hardware and software. So, the scheduling process required automation.

Considering the large potential for increasing operational productivity and the cost reduction involved in improving the quality of the scheduling process, the U.S. Coast Guard Research and Development Center embarked on a research project two years ago to automate various scheduling functions. The general goal was to build prototypes for each scheduling authority in which the computer system worked independently of explicit direction to help the user in scheduling a variety of platforms. Two prototypes of this nature have been installed in both Area offices which are the scheduling authorities for the high and medium endurance cutters and all the aircraft assets. The prototypes were designed in an evolutionary manner according to user requirements.

This paper will discuss a joint effort by the U.S. Coast Guard Research & Development Center, Idaho National Engineering Laboratory and Brunel University to provide the necessary tools to increase the human scheduler's capability to handle the scheduling process more efficiently and effectively. Automating the scheduling process required a system that could "think" independently of the scheduler, that is, the system needed its own control mechanism and knowledge base. Further, automated schedule generation became a design requirement and sophisticated algorithms were formulated to solve a complex combinatorial problem. In short, the resulting design

can be viewed as a hybrid knowledged-based/mathematical programming application system.

There has been much work carried out in intelligent scheduling, particularly in the area of production scheduling for job shops and flow shops [4], [11]. In a similar approach, Minton et al. [8] have adapted neural network techniques as a heuristic repair method within constraint satisfaction solvers. Dhar and Ranganathan [3] compare the performance of integer programming and expert systems approaches on a faculty course scheduling application. In the area of critiquing, the knowledge-based schedule diagnosis module of the system described here can be compared to that as set out by Silverman and Mezher [10], in that it functions as a passive, batch, after-task critique. In this paper, the expert system approach has been combined with integer programming to create an integrated system for intelligent scheduling. The diagnostic tools for validating a schedule are used in an integer programming framework to generate possible schedules and derive the "best" fleet schedule. By taking this approach for any given derived schedule, a report is produced for each cutter, detailing whether rules were broken and highlighting inconsistencies with the scheduling request.

Section 2. below examines the design for an intelligent platform scheduling support system which is expected to be operational early next year. Next, section 3 describes a generalized set partitioning model which is being used to automatically generate "good" schedules for USCG Area cutters. This model has been fully tested and "fine tuned" to produce timely schedules for implementation in the scheduling support system within the next few months. Although a mathematical program lies at the heart of the system, the model and the surrounding user interface have been designed in such a way that the user needs no optimization expertise. For instance, a screen-based user modeling interface allows the scheduler to specify new scheduling requests by either amending an existing scheduling request stored in the database or by creating a new schedule. Section 4 describes the use of expert system technology to analyze generated schedules since the mathematical formulation cannot capture all the rules that satisfy all the scheduling requirements. In Section 5 a particular application, the Cutter Scheduling Assistant Program (CSAP), which is being used in both Area offices to schedule large cutters is discussed. In particular, unique features such as the graphics-oriented direct manipulation interface and expert system are shown to be invaluable components for both usability and accuracy of

schedule outputs. Finally, in Section 6, possible directions for further research are presented.

2. An overview of the integrated system

In the past, schedules were manually created. This was a very time consuming and difficult task. The scheduler had to be fully aware of the rules of operating cutters, while trying to construct a schedule to meet the patrolling requests. After creating an initial schedule for all the cutters, it was generally the case that rules would be broken for some cutters and much work had to be carried out in validating the schedule. Thus the initial motivation was to automate the task of identifying faults in the user created schedule. In providing such a tool, it then became apparent that, while the work of the scheduler had been eased, the creation of a working schedule was still onorous. As a result there was a strong desire for further computer assistance. With the realisation of the schedule generator, a schedule is generated and fine-tuned manually. Now that the task of creating a schedule has become much easier, various scenarios can now be run to investigate the effect of policy decisions such as lengthening the time between maintenance or changing patrol lengths and the rules governing patrols.

Figure 1. below is an overview of the integrated system. The Relational Data Base System (RDBS) contains platform information, port information, historical assignments, planned schedules and task requirements. To protect the integrity of schedule data all transactions and queries must pass through the 'Data Conversion and Validation' bus. This will prevent existing schedules from becoming corrupted during analysis studies. The bus also controls access to all records of schedule changes to be used for planning purposes.

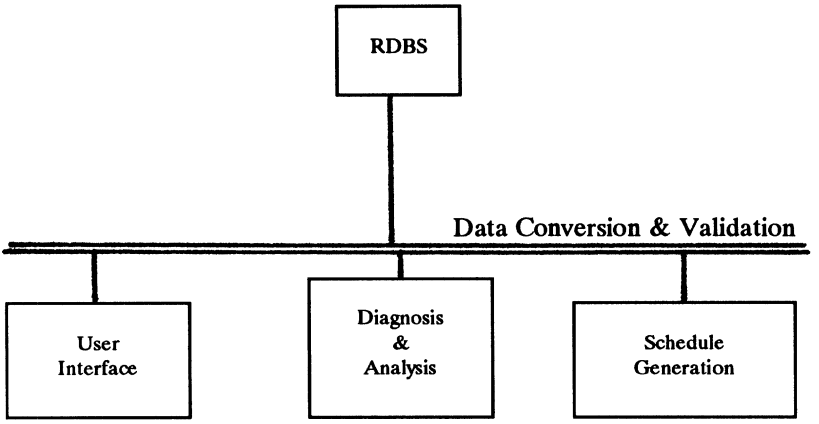


Figure 1. Overview of Integrated System

The interface bus provides a mechanism for transfer of schedule data between (bi-directional) scheduling support modules such as the RDBS and the 'Diagnosis & Analysis' module. The most typical usage pattern is: furnish state description information to the 'Diagnosis & Analysis' module, create and change assignments in a schedule, provide description of the modified schedule to the 'User Interface' module, load, display and (optionally) store the changed schedule in RDBS.

The 'User Interface' module is designed to be mouse-controlled and menu driven, with elements of graphics-oriented direct manipulation. The menu system allows a novice user to start using the system quickly, eliminating syntax errors or the need to memorize the commands. An example of direct manipulation is choosing a patrol location to assign a given platform. The user positions the cursor on the desired platform's timeline and drags the mouse for the desired patrol duration.

The 'Schedule Generation' module constructs the scheduling model and performs analyses of the solution output produced by the optimization module. The module's routines check the solution for feasibility and send updates to the 'Data Conversion & Validation' bus for possible inclusion as a planned schedule in the Data Base. The scheduling model is constructed by generating, for each platform, a set of possible schedules. The operational rules that govern the tasks and the platform duties are obtained from the Data Base through the 'Data Conversion

& Validation' bus. Finally, the 'Schedule Generation' module uses an optimization package to determine the "best" fleet schedule by selecting one of the possible schedules. The optimizer is specially customized to solve a generalized partitioning integer program efficiently. The next section fully describes this approach for a specific platform application.

In order to simplify the communication between modules developed in different locations by different organisations, a flat ASCII file format is used for the data interface. The content of the interface file fully describes the state of the schedule. This interface will not include all of the information known to the system, but only that which is considered necessary for the automated schedule construction program.
The following general types of information are represented: starting date for known schedule, description of each platform involved in the schedule, description of each activity (type of assignment) involved in the schedule, description of each assignment in the schedule, description of historical assignments occurring before the schedule start date, and description of each physical location relevant to the schedule.

A platform is described by the following information; name, class (type) of the platform, current location, future location, commissioning date and decommissioning date. An activity is described by the following attributes; name and a location (one or more occurrences) where this activity can take place. An assignment can be thought of as a relation between a platform and an activity for a specified period. It is described by platform name, activity name, start date, end date, location name (the location where this assignment will take place), statistical information such as the number of days underway, and the start date of the last assignment of a platform to a particular activity prior to the start date of the current context.

Finally, the 'User Interface' module graphically displays the "planned schedules" on a large color monitor in Gantt chart form. Using rules obtained from experienced schedulers, the module performs various diagnostics to critique generated schedules. In addition, the module contains software to edit a "planned schedule". The software can be used to enter the entire plan manually or to update an existing or computer generated schedule.

3. Discrete optimization model for scheduling

Both the 'User Interface' module and the 'Diagnosis and Analysis' module substantially reduce the difficulty in creating and maintaining cutter schedules. However, to adjust the basic input repeatedly in the reactive mode or to examine proposed changes to policy requires generating new schedules to test out the concepts. Manual generation is time consuming and would demand an unacceptable amount of professional staff hours. Therefore, a model was needed to generate realistic 'good' schedules for the 'Schedule Generation' module.

As discussed in section 1, the tasks assigned to the cutters are varied and include patrolling in specified localities of the area, training exercises and maintenance. The feasibility of a particular cutter schedule is governed by a set of operational rules that depend, in part, on the timing and nature of the tasks already assigned to the cutter before the start of the scheduling year in question. Other factors include transit allowances before and after a task, the duration of in-port time after the completion of a task and cutter capabilities. The requirements placed on the fleet fall into two principal categories. The first relates to minimum levels of coverage in terms of the number of cutters of given classes on patrol in each category at any given time. The second category of requirement concerns training and cutter maintenance. In these cases upper limits are placed on the number of cutters performing these tasks at any given time.

One approach to modeling scheduling problems of this type is to generate, for each cutter, a set of possible schedules, and to determine the "best" fleet schedule by selecting one of the possible schedules for each cutter. This formulation leads to an integer programming model which has been widely advocated (e.g. [5], [7]) in various guises.

However, in many scheduling applications, especially in the area of vehicle scheduling, the list of "requirements" often results in the non-existence of a feasible solution. In such cases, a relaxation of the requirements is necessary to obtain a schedule. On further examination of the practical issues it is frequently the case that some of the "requirements" merely reflect desirable characteristics rather than strict requirements. In developing a scheduling model that yields useful solutions, Darby-Dowman and Mitra [2] proposed the extended set partitioning model which was essentially an integer goal

programming formulation of a set partitioning model and admitted set covering, set packing and set partitioning as special cases.

In that model, the requirements were treated as targets and undercover (below target) and overcover (above target) were allowed but penalized. Our approach to the cutter scheduling problem follows a similar vein.

Within the operational rules that govern the tasks and the cutter duties, a set of possible schedules is generated for each cutter. An optimal schedule is one which is as close to meeting requirements as is possible. The generic model is formulated as minimize

$$\sum_{i=1}^{ng} \sum_{j=1}^{nt} \left(w_{ij}^{-} u_{ij} + w_{ij}^{+} o_{ij} \right)$$

subject to

$$\sum_{k=1}^{nc} \sum_{s=1}^{n_k} a_{ijks}\, x_{ks} + u_{ij} - o_{ij} = r_{ij}, \quad \begin{matrix} i = 1,2...ng \\ j = 1,2...nt \end{matrix}$$

$$\sum_{s=1}^{n_k} x_{ks} = 1, \qquad k = 1,2...nc$$

where

Parameters

nc = number of cutters to be scheduled

n_k = number of columns (possible schedules) for cutter k, k = 1,2,...nc

nt = number of time periods in the scheduling year

ng = number of constraint groups (schedule requirements)

Index sets

i = schedule requirements, i = 1,2,...ng

j = time period, j = 1,2,...nt

k = cutter identifier, k = 1,...nc

s = identifier of possible schedule for cutter k, s = $1,...n_k$

Model Variables

x_{ks} = 1 If possible schedule s for cutter k is selected

0 otherwise

u_{ij} = Extent of the under-achievement in respect of schedule requirement i in time period j

o_{ij} = Extent of the over-achievement in respect of schedule requirement i in time period j

Model Coefficients

a_{ijks} = 1 If possible cutter schedule s for cutter k contributes to schedule requirement i in time period j.

0 otherwise

r_{ij} = target/limit/threshold for schedule requirement i in time period j in terms of number of cutters contributing to the schedule requirement.

w_{ij}^- = Penalty for each unit of under-achievement in respect of r_{ij}

w_{ij}^+ = Penalty for each unit of over-achievement in respect of r_{ij}

The model is stated in a generic form. In any given application instance, simplifications may be possible. For example, if schedule requirement i is such that r_{ij} represents a desired lower limit then the penalty for over-achievement, w^+_{ij}, can be set at zero and o_{ij} considered as a logical variable. Similarly, if

schedule requirement i is such that r_{ij} represents a desired upper limit then the penalty for under-achievement, w^-_{ij} can be set at zero and u_{ij} considered as a logical variable.

The number of time periods in the scheduling year is a matter for judgement in relation to the individual application. A day-to-day schedule covering one year is required. Thus in its simplest form, nt equals 365 or 366. However, model size and hence solution time can be reduced by considering a larger time unit. In certain cases this can be achieved without loss of model validity. For example, if the duration of tasks and activities is always an integral number of weeks, the problem can be modelled on a week-to-week basis with n_t reduced to 53. Even if the duration of tasks and activities is not always an integral number of weeks it may still be worthwhile to adopt the time unit change to obtain solutions more quickly with a possible sacrifice on solution quality.

In this application, the approach taken is to generate a daily model and convert it to a weekly model which is then solved. The optimal weekly schedule is then matched with the columns of the daily model to obtain the daily schedule. A post processor that determines whether beneficial time shifts within weeks can be made to tasks is then applied. The daily to weekly conversion adopts a "conservative" approach such that the weekly solution results in an upper bound on the penalty associated with the schedule. Thus, if a constraint group i is a "minimum cover" group,

a_{ijks} = 1 if the possible (daily) schedule s for cutter k covers <u>all 7 days</u> of week j.
0 otherwise

However, if constraint group i is an upper limit group,

a_{ijks} = 1 if the possible (daily) schedule s for cutter k, covers <u>any day</u> of week j.
0 otherwise.

Although schedules are obtained using an integer programming model, the user runs the system routinely without any specialized mathematical programming knowledge. To achieve this, a robust model that can be reliably and predictably solved is needed. The structure of the model is such that an integer feasible solution is guaranteed since undercover and overcover are allowed. Solving the model in a predictable time has

implications on model size and solution strategy. The number of rows in the model is constant in the medium term and will only change if, for example, additional cutters are allocated to the area or a major redefinition of localities within the area takes place. The number of columns is governed by the number of potential cutter schedules generated. Experience has shown that a stratified random sampling from the full set of possible schedules for each cutter works well. In practice, models with about 100 possible schedules per cutter can be solved in about 15 minutes on a 80486 machine and little improvement in schedule quality is achieved by increasing the choice. There are many near equivalent schedules for a cutter and the stratification distinguishes between cutter schedules having similar structures. Such a process ensures a good cross section of cutter schedules in the model. Regarding the solution strategy, a full branch and bound search is impractical to implement on models of this size. The following simple strategy is highly effective.

Step 1: Solve the LP relaxation. If integer feasible go to END.

Step 2: Fix the cutter schedule variable with the largest value to 1. (Fix the other schedule variables for that cutter to 0).

Go to step 1.

END: Report solution.

This strategy follows a single path of a branch and bound tree to determine a near-optimal integer feasible solution. Although this approach would be impractical for general integer programming problems, the robust nature of the scheduling model ensures that the approach is viable. More sophisticated strategies including a three way branching process at each level of the tree were investigated. These strategies did not result in a significant improvement in solution quality but did require substantial increases in solution time.

The size of the models when the number of possible cutter schedules is restricted to a maximum of 150 per cutter is shown below.

	Model	
	Daily	**Weekly**
Number of constraints	3324	507
Number of variables x_{ks}	4156	4156
Number of variables u_{ij},o_{ij}	3294	477
Average number of non-zeros per x_{ks} column	240	30

Weekly models of this size can be routinely solved in 20-30 minutes on a 486 (33MHz) machine using the optimizer FortLP developed at Brunel University.

4. Schedule diagnosis and analysis

The use of heuristic methods for schedule construction, that is, in the form of expert system rules, was initially considered as a parallel effort to the development of a mathematical programming formulation. It became rapidly apparent that, in the cutter application, the mathematical programming approach would yield optimal (or near-optimal) schedules within the limits of available computing resources. Since heuristic methods were unable to provide guaranteed optimal solutions (or measures of suboptimality) and appeared almost as computationally expensive, we thus abandoned our investigation of heuristic methods for schedule construction.

The mathematical formulation cannot capture all the rules that satisfy all the scheduling requirements. Hence, the system has been given the capability to diagnose the generated schedules. Using rules which have been provided by experienced schedulers, the system will inform the user of potential problems within a particular cutter schedule. In this way, the system can produce coherent, complete and timely schedules for the major cutters.

As is typically the case in complex resource-constrained environments, it is virtually impossible to meet all constraints and goals when developing a cutter schedule either manually or automatically using a mathematical program. The critiquing capability helps to construct an operational schedule by

identifying outstanding commitments, and provides a critical evaluation of an existing schedule. The difficulty of creating and maintaining the schedule is largely reduced, particularly for officers rotating into new scheduling officer billets. Examples of problems detected by the schedule critic include:

- cutters overdue for maintenance or crew training
- insufficient number of cutters assigned to critical patrol localities.
- inappropriate assignment sequences, e.g., a training assignment followed by a maintenance period.
- cutters spending two consecutive Christmases on patrol.
- incorrect assignment durations.

The diagnostic program checks for activities that must occur on a periodic basis, e.g., training or maintenance, past due assignments, and constraints such as minimum or maximum durations for required assignments. In addition, the program must identify periods in which shortages occur in the targeted number of cutters assigned to a particular activity. Conversely, too many cutters may be assigned to the same activity at one time. In this case, the number of excess cutters and the durations involved must be reported.

Situations arise which require human intervention after schedule construction. For example, it is possible that the generated schedule could contain two contiguous, identical assignments. Human judgement would be required to decide whether or not to combine these assignments into a single assignment. Therefore, the diagnostic program must find all instances of this nature and allow the scheduler to make the appropriate modifications, if desired.

The mathematical program can handle dates or acceptable time frames for yards, docksides, training cycles, last years days away from home port, target days away from homeport, etc. However, certain special operations such as military exercises, cadet cruises and some types cooperative exercises cannot be accounted for during schedule construction. Common sense reasoning must be applied to many situations. For instance, trading in lesser capable cutters because of higher priority jobs for the more capable cutter at some location and time.

The needs of the crews are reflected in workload balancing, maximum cruise lengths, minimum times to be spent in homeport, and other items that affect the crew's "morale". These morale-related limitations result in goals that the scheduler tries to satisfy. The evaluation and comparison of two different schedules that satisfy the mission requirements and morale-related goals in different balances can not be performed using strictly objective criteria. Judgement and individual preferences are involved.

In the early stages of this project a rule-based expert system shell was used to perform the diagnosis of generated schedules. However, to improve performance of the critic, the rule-based evaluation was converted to a procedural form in the LISP code. The resulting loss of flexibility (for adding or deleting evaluation rules) was offset by a more than 90% reduction in the time required to complete a critical evaluation of the schedule. This experience is typical in a rapid-prototyping project: as system requirements become well-defined after user experience through several prototype evolutions, implementers can sacrifice design flexibility for more carefully tuned designs yielding higher performance.

5. Discussion

The Cutter Scheduling Assistant Program (CSAP) is providing Coast Guard schedulers with a powerful software tool to expedite the cutter scheduling process. In particular, cutters are assigned to various localities within an area to perform a variety of missions. The cutters are subject to availability of yards, docksides, and other maintenance facilities or requests such as "show and tell" for special occasions, e.g., holiday festivities and patriotic meetings. Many constraints are imposed on the schedule, such as military exercises, refresher training, long cruises, etc.

Each cutter's capabilities can be quite different and there are certain dependencies that exist among cutters that prevent treating this problem as a simple scheduling problem assigning independent, identical cutters to independent, identical tasks. For instance, one cutter may be preferred over another due to its special capabilities. Another complication arises because certain cutters are to be allocated together in a given place at a given time based upon their ability to cooperate in the performance of some mission task(s). Moreover, the scheduler must try to balance the workload democratically, e.g., all winter

patrols for one crew and holiday weather for others would be considered unfair to all but the most hearty sailor.

In a reactive scheduling mode, CSAP helps in the design of a robust (reliable) cutter schedule, as opposed to one that could be considered optimal according to naive criteria. CSAP starts with a given cutter schedule and tests its feasibility; if a given set of schedules is found to be infeasible, the system offers interactive or automatic procedures to change the given schedules until they are feasible. The detailed problems may have occurred for several reasons such as incompatible requirements specifiction, manual amendments, etc. If an individual problem is considered minor then the schedule may well be acceptable operationally since it is frequently impossible to satisfy each requirement, request, and desire, simultaneously.

CSAP automatically collects statistical data for the number of days away from home port and underway days for each cutter. In addition, the total number of cutters assigned to a particular district each day is automatically updated and displayed to the user as assignments are added or deleted. Soon, an ad-hoc query capability will allow the user to gather statistical data and other information that is not accounted for in the hard-coded CSAP graphical interface, but is contained within the data stored as a result of building cutter schedules with CSAP.

For the presentation of output, we continued the Coast Guard tradition of using the well-known Gantt chart to represent cutter assignments. The current CSAP prototype runs on an IBM compatible personal computer requiring at least 12MB extended memory and 1MB of free space on the hard disk. Additional capabilities include a variety of schedule hardcopy options, statistical reports and displays, daily accounting of cutters assigned to critical patrol areas, and alternate views of the schedule. Coding of the interface was substantially accelerated using an expert system graphics interface. The expert system shell provided a library of menu building and control routines that can be called from LISP.

In practice it is impossible to produce an overall fleet schedule that is ideal in every respect since desires and requirements frequently conflict. Additionally there is a level of uncertainty surrounding future requirements. The need to amend schedules, sometime by breaking rules, is therefore important and is addressed by the interface between scheduler and system. The critiquing module recognises occasions when rules are

broken and other aspects of the schedule which are not ideal, and, when requested, will report on these issues. However, such amendments are minor in comparison with the task of manually producing the entire schedule. By tackling the bulk of the work, the system allows the scheduler to concentrate on more important issues which can get swamped by the complexity of manual scheduling.

CSAP is being used routinely by the U.S. Coast Guard to schedule several large cutters in both the east and west coasts. Significant reductions in professional staff hours and the higher quality of the schedules produced using the system have provided ample evidence that CSAP will soon pay for its development costs. In addition, CSAP is a general-purpose system that can be used along with analytical models to help with the detailed investigation of alternative operating policies thereby providing a valuable planning and research tool for strategic analysis. By limiting a great deal of the 'experimentation' to the computer, it has the potential to reduce the cost of implementing new strategies, both direct dollar cost and indirect cost, involving Coast Guard personnel and the public.

Although the current CSAP prototype is designed specifically for cutter scheduling at the fleet level, much of it is directly applicable to similar resource scheduling applications, such as district level scheduling of cutters and patrol boats, aircraft, and crews. This flexibility has been achieved through an object-oriented design/implementation that decouples the display manipulation of objects (cutters, assignments, etc) from their internal representation for procedural manipulation.

As the CSAP system moved from the laboratory to become a fielded prototype, performance (response time) has become an issue of increasing importance. This "need for speed" was most readily apparent in the earlier versions of the rule-based schedule critic. For example, the earliest versions required over an hour to critique a one-year schedule. Although this was not regarded as a critical deficiency (pun intended), the slow turnaround adversely affected the prototype's usefulness as a tool for the "what-if" tasks essential in a decision support role.

6. Directions for further research

The CSAP prototype has been fielded since late 1990 as a

standalone monolithic system (in several versions). Current (1992) prototyping efforts are focused on moving the schedule data out of CSAP and into a relational database system. This will provide more flexible access to the data, such as for ad hoc queries, customized text reports, an interface to the schedule generation and optimization modules, and potential future extensions.

Even with the improved execution speed gained over several prototype evolutions, the current (mid-1992) CSAP prototype provides only marginally acceptable response times. The CSAP hardware platform (80386-based personal computer) and software environment (MS-DOS, Microsoft Windows, and expert system shell) all present various constraints on improving this performance. We expect these constraints to become increasingly binding as we incorporate database links and the aforementioned schedule generation modules into CSAP. Accordingly, the Coast Guard is investigating more suitable environments (e.g., RISC-based UNIX workstations, XII or Motif windowing systems, Common LISP) for a next-generation version of the CSAP prototype.

The overall system can be viewed as a hybrid knowledge-based and mathematical programming application system. The model generator incorporates not only the strict rules governing schedule acceptability but also other knowledge that is used to restrict the number of possible cutter schedules included in the model. The result is a model containing columns of good potential. The optimizer then acts as an efficient search procedure for schedule selection. Finally, the approach has the important virtue of being robust and is widely applicable. Extensions to other scheduling problems are now under investigation and development.

In summary, CSAP, incorporating the most important of the mission requirements and morale-related goals, is generating potential working schedules using a reasonable amount of computer resources. These potential schedules present the decision makers and schedulers with more choices for objective and subjective evaluation than the old manual method. Soon, this managerial tool will be applied to strategic and tactical decision processes, and is already improving these processes by relieving a heavy clerical workload.

7. References

1. Darby-Dowman, K., Hajian,M., Lucas,C., Mitra,G., and Smith, J., Solution Strategies for Zero-one Programming with a Class of Vehicle Scheduling Problem, Presented at EURO XII/TIMS XXXI, Helsinki, Finland, 29 June - 1 July 1992.

2. Darby-Dowman,K., and Mitra,G., An Extension of Set Partitioning with Application to Scheduling Problems, European Journal of Operational Research, Vol.21, pp.200-205, 1985.

3. Dhar, V., and Ranganathan, N., Integer Programming vs. Expert Systems: An Experimental Comparison, Communications of the ACM, 33(3), 323-336, 1990.

4. Fox, M.S., Constraint-Directed Search: A Case Study of Job Shop Scheduling, Morgan Kaufmann, San Mateo, California, 1987.

5. Hoffman,K.L., and Padberg,M., Solving Airline Crew Scheduling Problems by Branch and Cut, Working Paper, George Mason University, April 1992.

6. Kingsley,L., and Smith,J., A Scheduling Support System for the U.S. Coast Guard, In proceedings of conference on Artificial Intelligence Applications for Military Logistics, American Defense Preparedness Association, Fort Magruder Inn, Williamsburg, Virginia, 27-30 March 1990.

7. Marsten,R.E., and Shepardson,F., Exact Solution of Crew Scheduling Problems using the Set Partitioning Model: Recent Successful Applications, Networks Vol.II, pp.165-177, 1981.

8. Minton, S., Johnston, M.D., Philips, A.B., and Laird, P., Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method, Proceedings Eighth National Conference on Artificial Intelligence (AAAI-90), Boston, Massachusetts, July 1990.

9. Mitra,G., Darby-Dowman,K., Lucas,C., Moody,S., and Smith,J., Extended Covering and Partitioning Models for Scheduling in Industry, presented at the 1992 IBM Europe Institute, Oberlech, Austria, 3-7 August 1992.

10. Silverman, B.G., and Mezher, T.M., Expert Critics in Engineering Design: Lessons Learned and Research Needs, AI Magazine 13(1), 45-62, 1992.

11. Smith, S.F., Fox, M.S., and Ow, P.S., Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems, AI Magazine 7(4), 45-61, 1986.

About the Authors

Wesley Barnes is the Cullen Trust for Higher Education Endowed Professor in Engineering No. 6 at the University of Texas at Austin. Prior to joining the University of Texas in 1974, he was employed by Tracor, Inc. and Bell Telephone Laboratories. He is a past Associate Editor of the Transactions of the Institute of Industrial Engineers and is a Registered Professional Engineer in the state of Texas. He is very active in graduate research and supervision and possesses extensive industrial and consulting experience. He is the author of several books, among them are Network Flow Programming, Winner of Institute of Industrial Engineers Book-of-the-Year Award for 1980 and Statistical Analysis for Engineers and Scientists, A Computer-Based Approach, Revised Edition, McGraw-Hill Publishing Co., forthcoming, 1994.

E. Andrew Boyd is an Assistant Professor of Industrial Engineering at Texas A&M University. His research areas are primarily in the area of solution methodologies for large-scale integer programming problems. Under an FAA grant, he is evaluating traffic flow management models and solving large-scale integer programming problems. Dr. Boyd received a B.A. from Oberlin College and a Ph.D. from MIT.

Donald Brown received his B.S. degree from the United States Military Academy, West Point, the M.S. and M.Eng. degrees from the University of California, Berkeley and the Ph.D. degree from the University of Michigan, Ann Arbor. Dr. Brown is currently Associate Professor of Systems Engineering and Associate Director of the Institute for Parallel Computation, University of Virginia. He is a past Chair of the Operations Research Society of America Technical Section on Artificial Intelligence and a Vice President of the IEEE Systems, Man, and Cybernetics Society. His research focuses on decision and design aiding systems.

Rusty Burlingame holds a B.A. from Rice University. Mr. Burlingame is currently a graduate student in Industrial Engineering at Texas A&M University. His research interests are in integer programming, particularly scheduling systems. Mr. Burlingame also works on a part-time on-call basis for the MITRE Corporation's Center for Advanced Aviation System Development. In this capacity, he has developed custom software to solve 0-1 integer programming problems.

John Buzacott was born in Sydney, Australia. He obtained a Ph.D. from the University of Birmingham in 1967 and he has taught at the Universities of Toronto and Waterloo. He has been at York University since 1991. His research interests are in manufacturing system modeling, production planning and inventory control and

manufacturing strategy. He was President of the Canadian Operational Research Society in 1983-84 and is currently Chair of the ORSA Technical Section on Manufacturing and Operations Management.

Ken Darby-Dowman is a lecturer in Operational Research at Brunel University. He gained his B.S. degree in Statistics with first class honors from the University of Bradford, and M.S. and Ph.D. in Operational Research from Brunel University. His research interests center on modeling and algorithms for integer programming with particular reference to scheduling applications. He is currently research director of major funded projects on planning and scheduling in public transport and on dealing with uncertainty in optimization models for agricultural systems.

Raymond Fink is a Senior Engineering Specialist at the Idaho National Engineering Laboratory, Idaho Falls, Idaho. He received the M.S. degree in Engineering from Purdue University. His primary interests are in the development of intelligent support systems for complex task environments.

Kevin Gary is currently a software engineer working in signal processing at Global Associates Ltd. in Arlington, Virginia. He received his M.S. degree in Computer Science from Arizona State University in 1993 and a B.S. degree in Computer Science and Applied Mathematics from the State University of New York at Albany. He is a member of the American Association for Artificial Intelligence, and was elected to the Upsilon Pi Epsilon National Computer Science Honor Society in 1993. His research interests include knowledge-based scheduling, parallel search algorithms, and commonsense reasoning.

Fred Glover is the US West Chair in System Science and Research Director of the Center for Applied Artificial Intelligence at the University of Colorado, Boulder. He has authored or co-authored more than one hundred ninety published articles in the fields of mathematical optimization, computer science and artificial intelligence. Professor Glover is the first US West Distinguished Fellow, and is also an honorary fellow and reward recipient of several national societies and research organizations, including the American Association for the Advancement of Science, Alpha Iota Delta, NATO, IBM, the Energy Research Institute, the Decision Sciences Institute, the Institute of Management Sciences College of Practice and Miller Institute for Basic Research in Science. He has served on the boards of directors of several companies and consults widely for industry and government.

Othar Hansson is a founder of Heuristicrats Research, Inc. (HRI), in Berkeley, CA where he is currently Vice President, Research. He is completing his Ph.D. in Computer Science at the University of California, Berkeley. He earned his A.B. in English Literature and

Computer Science at Columbia College, and received his M.S. in Computer Science from UCLA, where he was a Rand Corporation Fellow. At HRI, he leads the DTS research and development team, and serves as a consultant to government and industry in the areas of artificial intelligence and software development.

Karl Kempf is a Principal Scientist in the Knowledge Application Laboratory at Intel Corporation in Phoenix. Arizona. He was the founding co-chairman of the AAAI Special Interest Group in Manufacturing (SIGMAN), and serves on the editorial board of IEEE Expert. He received a B.S. in Chemistry and a B.A. in Physics from Otterbein College, and did graduate work at Stanford University and the University of Akron while earning his Ph.D. in Computer Science. His research interests center on spatial and temporal reasoning systems applied to robots specifically and to manufacturing systems in general. He has published widely in these areas, in addition to presenting tutorials and workshops on these subjects at conferences and in invited presentations at universities and corporations. He has pursued these interests by designing, developing, and deploying artificially intelligent system in the field of Grand Prix motor racing with Team Ferrari (intelligent suspension design programs and track-side data interpretation systems), in the field of cinematic special effects with the Superman series of movies (intelligent robots to create the "flying" scenes), at McDonnell Douglas Corporation (intelligent robot programming systems), and at Intel Corporation (intelligent manufacturing scheduling systems).

Leonard Kingsley is an Operational Research Analyst with the Systems Analysis Branch of the United States Coast Guard's Research and Development Center. He has been at the Center since 1987. He was an invited guest speaker at the London School of Economics in April 1990 and has published numerous papers at ORSA/TIMS conferences. His background and interests are mainly in Computer Science and Business Administration.

Manuel Laguna is an Assistant Professor of Operations Management in the College of Business and Administration and Graduate School of Business Administration of the University of Colorado at Boulder. He received master's and doctoral degrees in Operations Research and Industrial Engineering from the University of Texas at Austin. He was the first US WEST postdoctoral fellow in the Graduate School of Business at the University of Colorado. He has done extensive research in the interface between artificial intelligence and operations research to develop solution methods for problems in the areas of production planning and inventory control, routing and network design in telecommunications, and vehicle routing. Dr. Laguna co-edited the Tabu Search volume of Annals of Operations Research. He is a full member of the Operations Research Society of America, the American

Society for Quality Control, and the International Honor Society Omega Rho.

Kenneth S. Lindsay is a member of the technical staff in the MITRE Corporation's Center for Advanced Aviation System Development. He is currently investigating the use of optimization modeling to support air traffic flow management. Prior to joining MITRE in 1991, Dr. Lindsay worked for the Magnavox Electronic Systems Company where he designed and developed software and provided computer security engineering for Department of Defense communication systems. Dr. Lindsay holds B.A. and M.S. degrees from the University of Pennsylvania, an M.A. from the University of Maryland, and a DSc from George Washington University.

Cormac Lucas is a lecturer in Operational Research at Brunel University. He gained his Ph.D. in Mathematical Programming Modeling from Brunel University in 1985. His research interests are principally in modeling and model analysis of linear programs. He has worked on industrial applications and is currently involved in one for the energy sector.

John A. Marin is an active duty Army Major currently pursuing a Ph.D. in Systems Engineering at the University of Virginia. He received a B.S. in Engineering from the United States Military Academy, West Point, NY, in 1979 and a Master's Degree in Operations Research from the Naval Postgraduate School, Monterey, CA, in 1986. His current research concerns intelligent decision systems and remote sensing.

Andrew Mayer is President of Heuristicrats Research, Inc. (HRI), in Berkeley, CA. In 1994, he received his Ph.D. in Computer Science from the University of California, Berkeley. In his research, he has focused on the development of new resource-bounded search techniques using probabilistic inference and decision-theoretic control, and the compilation of such techniques as a general method for designing efficient algorithms.

Kenneth McKay is Assistant Professor of Information Systems and Production Management at the Faculty of Business Administration, Memorial University of Newfoundland, St. John's, Newfoundland, Canada. He is interested in interdisciplinary approaches to the scheduling problem and how the human's understanding of the situation can be incorporated into model formulations. Dr. McKay is working on techniques for avoiding "dumb" scheduling decisions.

Gautam Mitra is Professor of Computational Optimization and Modeling and is the Head of Department of Mathematics and Statistics, Brunel University. He obtained his Ph.D. from London University in 1967 and has worked in the field of Applied

Mathematical Programming and Modeling for over thirty years both as an industrial consultant and as an academic researcher. He has published books with Academic Press, North Holland and Springer Verlag and has authored over sixty academic papers. He leads an active research group in Mathematical Programming based a the Mathematics Department, Brunel University.

Thomas E. Morton is Professor Emeritus of Manufacturing and Industrial Administration at the Graduate School of Industrial Administration at Carnegie Mellon University, and now partner in Parsifal Systems, Pittsburgh. He is a well known expert in forecasting, inventory planning, production scheduling and project management. Over the past 25 years, he has co-authored over 60 academic research papers on these subjects and has recently published a textbook on heuristic scheduling systems with extensive software for John Wiley and Sons. He is currently writing an Operations Management textbook for Southwest Publishers. He holds a B.S. degree in mathematics from the California Institute of Technology, and M.S., M.B.A. and Ph.D. degrees from the University of Chicago in mathematics, economics and operations research. His "Bottleneck Dynamics" scheduling approach with its use of accurate lead-time estimation and dynamic machine costing, is gaining acceptance as a cost-effective approach for both large and small implementations of scheduling systems for production and service job queues.

Sang Chan Park is an Assistant Professor of Information Systems Design and Analysis, School of Business at the University of Wisconsin, Madison. He received his B.B.A. (1984) from Seoul National University, Korea, and his M.B.A. in MIS (1985) from the University of Minnesota, Minneapolis. His Ph.D. in MIS is from University of Illinois, Urbana-Champaign. His research interest includes the application of machine learning in artificial intelligence to the design of knowledge-based DSS for such diversified areas as option investment, flexible manufacturing system scheduling, cellular manufacturing cell formation, and process planning. He has published several articles in the areas of intelligent process planning and knowledge-based scheduling.

Selwyn Piramuthu is an Assistant Professor of Decision and Information Sciences at the University of Florida. He earned his Ph.D. in MIS at the University of Illinois at Urbana-Champaign in 1992. He holds a B. Tech. from the Indian Institute of Technology, Madras, India, and an M.S. from the University of Arizona. His research and teaching interests are in machine learning, AI, human-computer interaction, database management, and simulation.

Narayan Raman is an Assistant Professor of Business Administration at the University of Illinois at Urbana-Champaign. He holds a B. Tech degree from the Indian Institute of Technology, a PGDM from Indian

Institute of Management and a Ph.D. from the University of Michigan. Dr. Raman's research and publications are in the areas of operation scheduling, line balancing and flexible manufacturing systems. He is a member of ORSA, TIMS and POMS.

Prasad Ramnath is a Post-Doctoral Fellow in Manufacturing and Operations Systems at the Graduate School of Industrial Administration at Carnegie Mellon University. His work includes research in large scale job shop and single machine scheduling, tabu search, beam search and other innovative search methods, and in understanding manufacturing systems/managerial objectives using statistical/econometric analyses. He holds a B. Tech degree in Mechanical Engineering from the Indian Institute of Technology, Madras, and M.S. and Ph.D. degrees from Carnegie Mellon University in Management of Manufacturing and Automation and Manufacturing and Operations Systems.

Frank Safayeni is an Associate Professor at the University of Waterloo, Waterloo, Ontario, Canada. He is interested in the organizational impacts of introducing new techniques and technologies to manufacturing, as well as to information processing activities of organizations. Over the past ten years, he has been involved in a number of field studies investigating how people perceive their work situation. Dr. Safayeni is exploring the organizational and conceptual issues related to the compatibility of JIT and CIM.

William Scherer received his Ph.D. degree from the University of Virginia in Systems Engineering, where he is currently an Associate Professor. He has authored and co-authored numerous publications on intelligent decision support systems, combinatorial optimization and stochastic control. His current research interests include: scheduling, decision analysis, intelligent systems, combinatorial optimization, and Markov decision processes. Dr. Scherer designed and built for the *International Telecommunications Satellite Organization* (INTELSAT) a simulation/optimization based scheduling and decision support system that incorporates expert rules into the simulation of a large-scale Markov decision process. Recently he has been working as a *Research Affiliate* with the *Jet Propulsion Laboratory* (JPL) designing and developing algorithms for the sequencing of activities for interplanetary unmanned spacecraft.

Michael J. Shaw is an Associate Professor of Information Systems at the Department of Business Administration, University of Illinois at Urbana-Champaign. He has also been a research faculty member at the Beckman Institute for Advanced Science and Technology, a research center on the same campus for the study of intelligence, where he is currently co-ordinating the research program in decision support systems. His research interests include machine learning, intelligent

manufacturing, distributed artificial intelligence, and knowledge-based decision support applications.

Stephen F. Smith is a Senior Research Scientist in the Robotics Institute at Carnegie Mellon University, where he is Director of the Intelligent Coordination and Logistics Laboratory. He holds a B.S. in Mathematics from Westminster College and M.S. and Ph.D. degrees in Computer Science from the University of Pittsburgh. From 1980 to 1982 he was Assistant Professor of Computer Science at the University of Southern Maine. He joined the faculty of the CMU Robotics Institute in 1982. For the past several years, Dr. Smith's research has focused on constraint-based reasoning frameworks and techniques for flexible planning, scheduling and control in practical domains. Dr. Smith has directed and participated in the development of several innovative knowledge-based scheduling systems, including the ISIS and OPIS manufacturing scheduling systems, the HSTS space observatory scheduler and the DITOPS crisis-action logistics system, and he has consulted on several other application development projects with various industrial and government organizations. His current research interests include mixed-initiative planning and scheduling in large-scale domains, reconfigurable and self-organizing production management systems, and agent-based modeling and analysis of supply chain dynamics. Dr. Smith has authored or co-authored over 60 articles in the area of knowledge-based planning, scheduling and control. He is a member of the ACM, the IEEE Computer Society and the American Association for Artificial Intelligence (AAAI), and is the academic co-chair of AAAI's Special Interest Group on Manufacturing.

Stephen P. Smith received the B.S., M.S. and Ph.D. degrees in Computer Science from Michigan State University, East Lansing, in 1977, 1979, and 1982, respectively. Dr. Smith has been working on software tools for scheduling and planning at Intel's corporate manufacturing group since 1991. From 1982 to 1991, he was a member of the research staff at Northrop's Research and Technology Center where he did research and system development on scheduling, image understanding, and groupware systems. Dr. Smith is a member of the Association for Computing Machinery, the IEEE Computer Society, the American Association for Artificial Intelligence, and Phi Kappa Phi. His research interests include: software and OO modeling tools for scheduling, simulation, and planning systems; the interaction of business and technical strategies on enterprise-wide performance goals; and the use of internetworking tools to enhance group performance.

J. Walter Smith (1938-1993) worked in the Systems Analysis Branch of the United States Coast Guard's Research and Development Center from 1972 until his untimely death on July 19, 1993. He had particular interests in mathematical programming and simulation and led many successful projects in these areas. A full member of the Operations

Research Society of America, Joe presented many papers at national and international conferences. His enthusiasm and vision are sadly missed by his many friends and colleagues.

Reha Uzsoy is an Assistant Professor in the School of Industrial Engineering at Purdue University. He holds a B.S. degree in Industrial Engineering from Bogazici University, Istanbul Turkey. He received his Ph.D. in 1990 from the University of Florida and joined the faculty of Purdue University the same year. His teaching and research interests are in production planning and scheduling, facility design and economic analysis, with particular applications in semiconductor manufacturing. Before coming to the U.S., he worked as a production engineer with ArcelikASA, a major appliance manufacturer. He has also worked as a visiting researcher at Intel Corporation.

Index