Solving Dynamic Sequencing Problems

David W. Hildum Daniel D. Corkill
Department of Computer and Information Science
University of Massachusetts

COINS Technical Report 90–63
July 26, 1990

Abstract

Sequencing problems describe a class of problems that require the assignment and ordering of a collection of resources for a set of jobs. The processing of a job consists of performing a sequence of tasks that require various resources.

We explore a class of problems we call *dynamic sequencing problems*, where the complete set of jobs is not always known in advance, not all jobs arrive on time, resources break down unexpectedly, and there is perhaps more than a single criterion for evaluating schedule quality. Furthermore, all of these events occur in real time.

We present the design for a problem-solving system that emphasizes the integration of multiple scheduling perspectives and the development of an opportunistic, least-commitment approach to solving dynamic sequencing problems. Both of these techniques help to produce a scheduling system that makes intelligent decisions based on the most current information that exists in the environment, and avoids unnecessary processing that results from making hasty or uninformed decisions that could be precluded by further analysis.

This research was sponsored in part by gifts from Texas Instruments, Inc., and by the National Science Foundation under CER Grant DCR-8500332. Machine resources were supported by the Office of Naval Research, under a University Research Initiative Grant (Contract N00014-86-K-0764).

1 Introduction

The problem of assigning and ordering a collection of resources to a set of jobs in order to satisfy a group of constraints is an instance of a class of problems called *sequencing problems* [Bellman *et al.*, 1982, Coffman, 1976]. Sequencing problems describe a variety of interesting situations, including job shop or factory scheduling, where the jobs are orders for products and the resources are the machines used to produce the products; the assignment of vehicles to routes (as in a public transportation environment), where the jobs are specific routes and the resources are buses, trains, or tracks; and the scheduling of programs on a computer, where the jobs are the programs, and the resources are the processors. The constraints for most of these problems include meeting the due dates for orders, minimizing the amount of time jobs spend being processed, and making optimal use of the resources.

In the classical sequencing problem, the overall goal is to produce for each of a set of orders, an explicit sequence of operations to be performed on members of a set of resources in order to satisfy some set of constraints. A specific result may be produced using a number of different operation and resource sequences, and a specific operation and resource sequence may be used to produce a number of different results. An order can be thought of as a request for the production or servicing of some quantity of product that is achieved through the execution of a process plan, process routing, or task sequence that includes all potential operations and resources in some particular order. A job is the sequence of processes, operations, or tasks used to produce or service the product requested by an order, and a machine or processor is a resource used to perform an operation.

The quality of a schedule (the collection of process plans for a set of orders) can be evaluated according to a number of factors, like the degree to which orders meet their due dates, the average amount of time that orders spend waiting for resources, and the amount of time that resources in the environment are idle.

Historically, attempts to solve sequencing problems have relied on applying linear programming techniques or severely restricting problem descriptions in order to optimize a single objective function (like minimizing flow time or make-span) [Bellman *et al.*, 1982, Coffman, 1976, French, 1982]. The problem is generally constrained to the point where each operation can only be performed by a single resource, each job includes every operation exactly once in its process plan, and all parameters—the number of jobs and their arrival times, the number of resources and the duration of processing times are known and remain fixed from the outset. These restrictions, however, exclude non-trivial sequencing problems of the kind that tend to occur in the real world.

The real world does indeed make things difficult. The number of jobs and their arrival times are *not* known in advance. The number of resources and the duration of processing times are *not* fixed. Furthermore, there is often a choice of more than one resource for performing a task, and more than one process plan for processing an order. In addition, time is a factor. As jobs enter the system, or when resources break down, the schedule must often be modified quickly so as not to delay current processing. These dynamic aspects contribute to the definition of a real-world class of sequencing problems that we shall call *dynamic sequencing problems*.

1.1 A Dynamic and Unpredictable Environment

Dynamic sequencing problems occur in dynamic environments. Even in situations where a schedule of orders is known and expected at the outset, orders can arrive ahead of, or behind schedule. Anticipated orders are canceled and new ones are added without notice.

While the shutting down of resources for the purpose of periodic maintenance, or their unavailability due to changing work shifts can be anticipated during scheduling, resources are susceptible to unexpected failure. In addition, as the flow of orders through the system and the availability of resources changes, bottleneck conditions (resulting from an inability of the system to meet its resource requests) develop and recede. For example, in an office environment, if one of two copying machines breaks down unexpectedly during lunch time, it may not cause a significant backup of copying jobs until more people have returned from lunch.

Process plans often must be altered in response to the occurrence of unpredictable events in the environment. In the public transportation domain, the breaking of a water main and subsequent flooding of various streets forces the use of a different set of streets (the incorporation of a new plan segment into the original process plan) for the affected bus routes. These unpredictable events can void the applicability of the process plans of jobs currently being processed and idle jobs waiting for execution. They also require the planner to be able to augment process plans with sub-plans that help alleviate failure situations and may require additional resources.

Finally, processing time is inherently dynamic and unpredictable in the real world. In a laundromat, some clothes driers take longer than others to dry a load of wash. Different resources require different amounts of time for performing the same operation. A difference in operators or even the particular time of day at which an operation is performed can affect the duration of the processing.

1.2 The Importance of Time

Dynamic sequencing problems are real-time problems. A system for solving these problems must make decisions for the future at the same time that its previous decisions are being executed and are possibly failing. Furthermore, the system may not be immediately informed about changes in the environment, and the implementation of its fixes may also be delayed. While the system stops to think and correct, the real world situation continues to deteriorate. Every moment of indecision complicates the problem at hand.

1.3 Multiple Planning and Scheduling Perspectives

The process of determining task sequences and securing resources for jobs in any kind of sequencing problem requires reasoning about the conflicts that arise in the attempt to satisfy the various constraints among the resources, tasks, and orders in a domain. Conflicts arise when the satisfaction of a particular constraint or constraints affects the ability to satisfy others. Since solving the sequencing problem requires the interleaving of typically separated planning and scheduling phases in the process of producing task sequences and schedules of resource usage, these conflicts arise even more frequently. In order for a problem-solver to intelligently reason about a conflict, it must consider the entire set of constraints that enter into the situation.

In sequencing problems, both jobs and resources have constraints that govern their manipulation for scheduling purposes. Each job can express a preference for certain tasks or a particular sequence of tasks. Resources can require periodic shutdown for maintenance purposes, or prefer that their usage be balanced and that drastic set-up changes be minimized. If a problem is organized in such a way as to focus on satisfying a limited group of constraints, then its ability to adequately address the conflicts that arise out of conflicts with other constraints is lessened [Smith and Ow, 1985].

If the assignment of resources is done solely from the viewpoint of each order separately, then the constraints on individual orders are implicitly given priority over the constraints of the set of orders, which are in turn given priority over the constraints on the resources. The result is sub-optimal resource allocation. Constraints such as balancing resource usage and minimizing resource set-up costs are sacrificed. If, however, the assignment of resources is done from the viewpoint of the resources themselves, then it is the constraints on the resources that receive priority. The result is that orders may miss their due dates or spend excessive amounts of time waiting for the use of particular resources. A system that properly integrates concern for all constraints into the planning and scheduling processes in an attempt to further understand the nature of the conflicts that will invariably occur is clearly desirable.

In dealing with this issue, Smith, Fox, and Ow [Smith *et al.*, 1986a] have identified the utility of first decomposing the problem into multiple perspectives—which they term *order-based* and *resource-based*—and using the order-based perspective for dealing with conflicts that center around orders (such as task sequencing conflicts) and the resource-based perspective for dealing with conflicts that center around resources (such as bottleneck situations)[1].

2 Past Approaches

In this section, we discuss the difficulty of solving classical sequencing problems, and describe the most closely related work on solving specific instances of the dynamic sequencing problem using AI techniques, namely the development of systems for solving job shop scheduling problems.

2.1 The Classical Sequencing Problem

The classical statement of the problem makes a number of strict assumptions about jobs, resources and processing times. These assumptions serve to highly constrain the classical sequencing problem. Basically, all required knowledge is assumed to be known and fixed from the outset. If there are n jobs each requiring o operations, then there are $r = o$ kinds of resources, and hence r total resources (one of each type). The goal is to produce a schedule that includes an explicit sequence of o operations for each of the n jobs. The only space for slack is provided by allowing resources to be idle during the make-span for the order set.

The number of feasible schedules that can be produced for a collection of n jobs each requiring o operations is $(n!)^o$. With even moderate values for n and o, this number gets very large. Relaxing the constraints to allow for a choice between two resources for each operation results in a total number of $(2^n n!)^o$ feasible schedules.

[1] Section 2.2 contains further discussion of this work.

Again, with moderate values for n and o, this number gets even larger. It is clear that sophisticated methods are necessary to help avoid considering each of the feasible schedules in an attempt to find a satisficing one given the optimization constraints.

Various linear programming techniques have been developed to determine optimal solutions for appropriately restricted sequencing problems, but they do not scale up to real-world situations that tend not to adhere to the classic assumptions, and often require solutions that are optimal according to a number of different perspectives. They also rely on enumeration methods that are infeasible in realistic time-constrained environments. Furthermore, linear programming techniques are intended to produce only initial schedules. They do not address the problem of modifying a schedule in response to developments in an unexpectedly changing environment.

2.2 Knowledge-Based Job Shop Scheduling

The relevant work in applying AI techniques to the sequencing problem has been done in the ISIS (Intelligent Scheduling and Information System) [Fox *et al.*, 1982, Fox, 1983, Fox and Smith, 1984] and OPIS (Opportunistic Intelligent Scheduler) [Smith and Ow, 1985, Ow and Smith, 1986, Smith *et al.*, 1986b] projects[2]. The ISIS work in particular focused on the representation and manipulation of a variety of scheduling constraints for use in constraining (narrowing) the search space of potential schedules in a job shop or factory scheduling environment.

The ISIS system uses a four-phase hierarchical approach to scheduling. It first selects from a set of orders the order with the highest priority, and then (in the second phase) analyzes that order's resource requirements to help constrain later searching during subsequent phases. The third phase develops a sequence of operations and resources for the order using a beam search approach that gradually expands a graph representing the order's process plan. Constraints are used heavily during this phase to produce candidate paths in the schedule graph, and to help rate and prune the branches in the graph for expansion by the beam search. The result of this step is an operation and resource sequence for an order that lacks only the specific resource assignments (processing time slots), which are secured in the fourth phase.

The major advantages of this work are its contributions to the area of constraint definition, and the use of constraints to narrow an enormous search space. The disadvantages center on the overall static nature of the control of the system, and its lack of flexibility, namely in its use of a beam search that cannot anticipate future scheduling problems. Furthermore, its sole focus on order-based constraints during planning and scheduling limits its ability to deal with important resource-based conflicts and inter-order constraints. It is also susceptible to the problem of not recognizing dead-end scheduling situations before a substantial amount of problem-solving work has already been done.

The OPIS project was designed with the intent of investigating the notion of *multiple problem decompositions* and incorporating an opportunistic approach into the ISIS scheduling mechanism [Smith and Ow, 1985]. Instead of viewing the scheduling problem as a collection of independent agents (shop orders) generating plans to satisfy their own goals and constraints (as in ISIS), the OPIS project suggests a decomposition of the problem that identifies two kinds of agents—orders and resources—each having

[2]See [Smith *et al.*, 1986a] for a discussion of the issues involved in both systems.

their own goals and constraints to satisfy. The idea is that resource allocation requires the consideration of a collection of resource-based constraints (like resource capacity limits and task sequencing preferences) that are effectively ignored by a problem decomposition that considers only order-based constraints. OPIS uses its order-based and resource-based scheduling perspectives to help improve its ability to deal with scheduling conflicts and to avoid the horizon problem that occurs in ISIS that precludes its being able to accurately predict future problems arising from resource shortage. In order to use multiple perspectives during scheduling, bottleneck resources are initially identified by the user. The resource-based perspective is then adopted to secure reservations for all of the bottleneck resources. The remaining scheduling is order-based as in ISIS, but with the bottleneck resource reservations serving as *islands of certainty* around which the beam search can expand potential schedules.

Later work on OPIS recognized that bottleneck conditions in a job shop are indeed dynamic, making an *a priori* designation of these resources of limited use. Also, it was determined that the assumption that the resource-based perspective took priority over the order-based perspective was not justified. In addition, the system was still not able to adequately react to the unexpected conditions that arose in the environment. In light of these developments, a blackboard architecture was adopted, and in order to increase the predictive and reactive capabilities of the system, two knowledge sources were added—one to predict bottleneck resource conditions by looking at all of the orders beforehand, and the other to help adjust a preliminary schedule (using a simple right-shifting approach) in the event of conflicts developing between the order-based and resource-based scheduling decisions. More recent work on OPIS has focused on the problem of reacting to constraint failures resulting from low resource capacity and inconsistent processing times for operations within an order's process plan, with the emphasis on producing corrected schedules of high quality [Ow *et al.*, 1988]. A problem with OPIS remains, however, in that it still cannot accurately predict future bottleneck conditions for the shop, because changes to the environment during schedule execution, like resource failures and order cancellations—which greatly affect bottlenecking—cannot be completely and accurately anticipated at the outset.

3 Our Approach

Our approach to extending the knowledge-based techniques pioneered by ISIS and OPIS involves:

- Integration and use of multiple perspectives in all phases of the planning and scheduling processes.
- A least-commitment planning approach that exploits the availability of and interdependencies among resources.
- An opportunistic control component for quickly reacting and adapting to a dynamic and unpredictable environment.
- An event-driven constraint failure and conflict resolution mechanism for intelligently handling planning and scheduling problems.
- A goal clustering component for representing and exploiting order and resource intentions.

The primary features of the approach are the integration of multiple scheduling perspectives and the development of an opportunistic, least-commitment approach to solving dynamic sequencing problems. Both of these features help to produce a scheduling system that makes intelligent decisions based on the most current information that exists in the environment, and avoids unnecessary processing that results from making hasty or uninformed decisions that could be precluded by further analysis of the problem.

Past approaches to solving this problem have not entirely addressed the complete set of issues that arise in dynamic sequencing domains, namely the limited reliability of initial expectations and projections based on the information available at the outset of the scheduling process. If the repair of scheduling conflicts in a dynamic environment is recognized as an important goal for intelligent scheduling systems, then equal weight should be placed on approaches that attempt to avoid the occurrence of those conflicts from the outset by maintaining current views of the problem-solving situation that incorporate the various perspectives of all the important agents.

In the remainder of this section, we elaborate on some of the techniques we are using to solve dynamic sequencing problems.

3.1 Integration of Multiple Scheduling Perspectives

The integration of multiple scheduling perspectives allows for decisions to be made based on information provided from a range of viewpoints. When conflicts arise, the various appropriate scheduling perspectives—those involved in the conflict and those affected by potential solutions—have a voice in deciding how to go about the process of recovery. The representation of the goals and constraints of both orders and resources plays an important role in this process. Accurate scheduling perspectives depend on representations of goals and constraints that help identify the important aspects of specific situations.

Previous approaches to solving dynamic sequencing problems that make use of multiple scheduling perspectives have relied on either a static *a priori* specification of bottleneck resource conditions or an initial analysis of the anticipated job schedule and resource availability estimates to force the consideration of a resource-based perspective into an otherwise order-based scheduling procedure [Smith *et al.*, 1986a]. We intend to fully incorporate the consideration of multiple scheduling perspectives into all phases of the planning and scheduling, and to do so using up-to-date information gleaned from the environment. We thus avoid the problems of using a resource-based perspective only when bottleneck resources are initially scheduled (as opposed to *whenever* they are scheduled, as in the case of resolving constraint conflicts or satisfying relaxed constraints), and having to organize our planning and scheduling processes according to the results of analyses that have little reliability once unpredicted changes begin to occur in a dynamic environment.

3.2 Least-Commitment Planning

The construction of process plans frequently results in backtracking and expensive constraint failure in response to unexpected changes in the environment. Most often, however, these problems can be avoided by using a least-commitment planning approach that delays making decisions for as long as possible, so that hasty uninformed

decisions are eliminated, and decision-making (when it does occur) is highly informed. The knowledge used to inform the decision-making comes out of a rich representation of the domain, including information about operations, resources, and their constraints. A hierarchy of operations can be inferred for each order based on the kinds of operations that are required, how they depend on one another, whether they have secured resources, and how scarce their required resources are during the times they are needed. This dynamic hierarchy provides the scheduler with valuable information for use in deciding what aspects of an order should be elaborated at a given moment in order to minimize its commitment and avoid producing conflicts between constraints.

A drawback of ISIS is its use of a beam search that relies on temporal sequencing constraints to guide its expansion of process plans for orders. Because it ignores potential information about operation and resource inter-dependencies, it runs the risk of devoting processing time to building infeasible process plans. If the last operation for a job requires a resource that is not available, then the beam search cannot detect this problem until it has nearly completely expanded the process plan and then attempts to find a resource for the problem operation. The least-commitment approach taken in our work is designed to avoid this problem and to take advantage of all potential information when making decisions so that rash unnecessary commitments are postponed and the need for backtracking is thus minimized. By knowing that a particular resource is scarce at a particular time, the system can solve that problem first and then move on to less constrained aspects of the problem. While OPIS is able to deal with bottleneck resources before expanding the rest of a process plan, neither ISIS nor OPIS can make use of the notion of dependency that arises in the case of vehicle and work site resources, where the work site resource should be secured before the vehicle resource.

3.3 Opportunistic Control

By integrating multiple scheduling perspectives together with an opportunistic approach to control, we avoid the need to limit the control options available to the planner and scheduler, and instead allow them to opportunistically focus on important aspects of the problem as they surface, regardless of their perspective. We thus allow the dynamic nature of the environment to dictate the aspects of the problem that should be addressed, and provide the system with the knowledge of how to decide among the actions that are warranted. This approach also provides the ability to schedule orders based on their priority and the time at which they are received, and to select during each problem-solving cycle an order for processing from among a group of partially completed orders. In addition, the completion of a process plan for an order can be organized such that the pressing details of the plan are determined early on while the less pressing details are not decided upon until later.

OPIS uses a limited opportunistic scheduling approach that expands partial process plans after bottleneck resources have been secured, but still treats the remaining planning and scheduling for an order as an atomic action. It displays some opportunism during the expansion of process plans, but nowhere else. We use opportunism in *all* facets of the planning and scheduling processes (such as process plan expansion, order selection, and constraint conflict resolution), in order to be able to deal intelligently with the dynamic nature of the real world. Opportunism provides the benefit of being able to react quickly and appropriately to the situation at hand regardless of

whatever task is currently being performed.

3.4 Event-Driven Constraint Failure and Conflict Resolution

Constraint failures and conflicts are triggered by decisions made elsewhere in an order's processing plan and in response to resource failure or unavailability. A single event may cause multiple constraints to fail. Though the least-commitment approach minimizes these failures, they do occur, and their resolution requires careful consideration. Our approach to conflict resolution and the correction of constraint failures uses knowledge of constraints, their interaction with each other, and their potential methods of relaxation to determine both how and when to repair their failures. Projection is used to see how local changes will affect other parts of a plan to provide further information to the knowledge source(s) that are stimulated to perform the necessary corrections. Given the opportunism inherent in the control of the scheduler, the correction of constraint failures and conflicts can be arranged to occur immediately or at a later time, depending on the seriousness of the conflict, the current status of the processing, and the nature of the correction. Relaxations of constraints are saved with an order's plan so that in the event that conditions arise such that a previously relaxed constraint can be better satisfied, steps can be taken to attempt to re-satisfy the original constraint.

An important issue here is the process of deciding *when* to go about implementing the appropriate conflict resolution activities, once the *how* has been determined. Previous sequencing systems (specifically OPIS) have focused on the process of deciding what actions to take in reaction to particular types of conflict, but they have not dealt with the question of exactly when these actions should be executed [Ow *et al.*, 1988]. We intend for the process of building this mechanism to bring out the issues involved in getting a system to make full use of its generic and domain knowledge to determine not only the proper sequence of actions to take in response to constraint failures and conflicts, but also the specific scheduling of the application of these actions.

3.5 Service Goal Clustering

The elaboration of a process plan for an order includes the satisfaction of requests for resources to perform various operations. These *service goals* stimulate knowledge sources that attempt to secure a requested resource during some specified time slot. An understanding of inter-order constraints allows us to reason about the network of service goals that exist during the scheduling of orders, and optimize the execution of knowledge sources to take advantage of this higher view. For example, consider a situation where a sequence of operations for different orders are to be performed at the same work site. Instead of scheduling a resource-securing knowledge source for each order, we can group the service orders together (recognizing that they are to be performed sequentially at the same site) and stimulate a single knowledge source to secure the required resource for the combined time slots of all the operations. The important consideration is the tradeoff between spending time clustering and spending time separately scheduling all resources. Goal networks provide valuable information about the current status of problem-solving by representing resource preferences and providing explicit global views of problem-solving that can be used to improve the sophistication of the system.

Previous sequencing systems have not made use of groupings or networks of goals that indicate requests for specific operations or resources. We believe that information about these intentions can be effectively used to make planning and scheduling decisions in groups instead of independently, and that this can be done with less processing for the system.

4 Airport-Based Airline Resource Management

In this research, we are focusing on the domain of airport-based airline resource management, that is, the scheduling of an airline's ground services at a particular airport in an attempt to minimize delays in servicing airplanes, and to maximize the efficient usage of resources in order to meet important flight deadlines. The penalties for producing bad schedules are expensive in terms of dissatisfied customers in a highly competitive business environment, and the creation of idle time for very expensive airplanes and other equipment.

The airline resource management domain requires a sophisticated planning and scheduling system to produce operation sequences and resource reservations that will satisfy a desired timetable. In addition, the system must be able to adapt to the developments in a dynamic environment that affect the ability to adhere to the desired timetable, and intelligently decide on appropriate actions to take in response to these situations in order to produce results that are as close to the original performance goals as is possible.

Recently, increasing attention has been paid towards the airline resource management domain by the AI community. GATES [Brazile and Swigger, 1988] is an expert system designed (and used) to perform airplane gate assignments for Trans World Airlines (TWA) at New York's John F. Kennedy International Airport (JFK). It performs two tasks—the generation of an initial schedule for all daily TWA flights to and from JFK, and the alteration of a given schedule in light of new constraints encountered in real time. Similarly, Texas Instruments is currently involved in at least two joint ventures (with Iberia Airlines and Air Canada) to develop software systems for managing a wide range of (not just airport-based) airline resource management issues, from the automation of airport ground service scheduling (and re-scheduling) to the scheduling of periodic heavy maintenance checks for aircraft.

Flight schedules are prepared periodically by airline companies. Gate and other resource capacities at airports are considered in this process. Once the daily flight schedules are determined for an airline at an airport, ground controllers assign gates and other resources to the flights in advance. When problems occur during the execution of a daily schedule, these ground controllers are required to repair the schedule on the fly so as to keep things running as smoothly as possible.

More recently, many airlines have begun leasing entire airport terminals at specific airports (called *hubs*) and routing many of their flights through them. The result is that many large airports now have an airline that uses a significantly large number of the airport's gates solely for its own flights. With passenger air travel steadily increasing and airport building and expansion becoming ever more difficult and expensive, the problem of scheduling increasing numbers of flights for a generally static collection of resources becomes even more difficult to solve.

The airline resource management domain is a prime example of a dynamic se-

quencing problem. The set of orders is represented by a schedule of flights that require ground servicing. The jobs are the sequences of tasks undertaken in the process of servicing flights, and the resources are represented by the equipment necessary for performing the servicing tasks. The airline resource management domain makes the same assumptions as the dynamic sequencing problem that contribute to making it a richer environment for experimenting with sophisticated planning and scheduling techniques.

5 The Current Implementation

In an attempt to explore the issues that arise in dynamic sequencing problems, we have built a system that simulates the airline resource management problem. The ARS (Airport Resource Scheduler) system is comprised of a knowledge-base that defines the aircraft and other resources necessary for the domain, and a collection of knowledge sources (the scheduling component) to secure resources for the operations that require them. The third component is a planner that performs a simple top-down (goal-directed) approach towards generating process plans for orders. The remaining component is an event-based simulator that generates (simulated) real-world events that initiate processing by the planner and scheduler.

The simulator requires two inputs: a description of an airport that includes the airplanes and other resources that are available to an airline for use in running and servicing flights, and a flight timetable for the airline created for a specific airport (ideally the same one loaded into the system).[3]

Orders are defined by descriptions of the operations and the potential sequences of those operations that must be performed in order to satisfy them. Operations are described in terms of the expected duration of time in which they can be completed (dependent on the situation), and the resources that they require. Resources are described by their speed in both getting to and actually performing specific operations.

The system is essentially divided into two separate subsystems—the event-based simulator and the combined planner and scheduler.

The simulator is responsible for initiating the events that occur in the airport domain—that is, local and external takeoffs and landings of arriving and departing flights, and the dispatching of assigned resources to their designated servicing jobs in response to commands issued by the scheduler.

The planning and scheduling modules together behave like a standard goal-directed problem-solver. The scheduler is comprised of the collection of knowledge sources that secure resources and are executed in response to the creation of goals requesting that specific service operations be performed on a flight. These (service) goals are created by the planner in response to a request for the servicing of a flight. Upon being notified of the existence of a flight to be serviced, a collection of these goals are created that correspond to the operations that are required to service the flight. The collection of service goals to be created for the flight are specified in the definition of the type of service that the flight requires (quick or long turnaround). The goals are first

[3]The body of knowledge describing the characteristics of the orders, operations, and resources specific to aircraft flight servicing is currently built into the system. In our desired system, this knowledge would serve as the third input to the simulator, and would be defined separately, using generic (domain-independent) order, operation, and resource concept definitions provided by the underlying system.

rated, after which they trigger knowledge sources to schedule the resources required to perform the specific requested operations (if resources are indeed required). An order's process plan is refined as resources are secured for the operations required to service the flight. When resources are assigned to a flight servicing operation by the particular knowledge source, an assignment message is issued to the simulator, which in turn causes them to be dispatched to the appropriate destination at (close to) the proper time. When all resources required for servicing a flight are secured and show up at the proper gate location in order to perform their respective operations, the flight gets properly serviced. At the moment, however, if any required resource cannot be secured, or the resource does not show up at the proper gate location, then all operations that can be done are performed, after which the flight remains (permanently) stalled (waiting for the remaining missing resources).

The flight timetable currently used in testing the system is the schedule of all daily Northwest Airlines (NWA) flights through Detroit, MI (one of NWA's hub airports). This timetable includes more than 4330 weekly flights, averaging over 600 per day. NWA leases 53 gates (5 concourses) of the Detroit Metropolitan Wayne County Airport.

The ARS system is written in Common Lisp and CLOS, and is implemented using GBB (the Generic Blackboard System) [Blackboard Technology Group, Inc., 1990].

6 The Future Design

In order to test our ideas, we have designed a system to intelligently plan operation sequences and resource schedules to solve dynamic sequencing problems. A major goal of the implementation is to develop a basic system upon which task-specific application programs can be built for studying a variety of dynamic sequencing problems. For the purposes of experimentation, we are also building a domain-specific application for managing airline resources. Figure 1 provides an idea of the basic layout of our proposed system. In the remainder of this subsection, we describe the specifics of the two main modules of the system. The simulator will remain just about as it is in ARS, with only minor alterations to make it more generalized.

6.1 Generic Concepts

The knowledge about operations and resources and how to plan operation sequences and secure resources for a set of orders must be fully separate from the knowledge about any specific domain. Concepts related to the dynamic sequencing problem must be defined and made available for use by specific domain implementations. By developing a generic semantics, we can provide the capability to implement a variety of dynamic sequencing domains.

The descriptions of operations, resources, and various kinds of constraints will be modeled after similar work by Fox [Fox, 1983]. The representation of resources will also make use of some of the techniques of Pease (specifically *resource models*) [Pease, 1978] to organize and provide the necessary knowledge about resources and their constraints for use by both the planner and the scheduler.

Orders can be thought of as representing requests for the production of some product. They specify the product, the quantities requested, expected ready times, and due dates. Products in our system are described by *task networks* that identify the

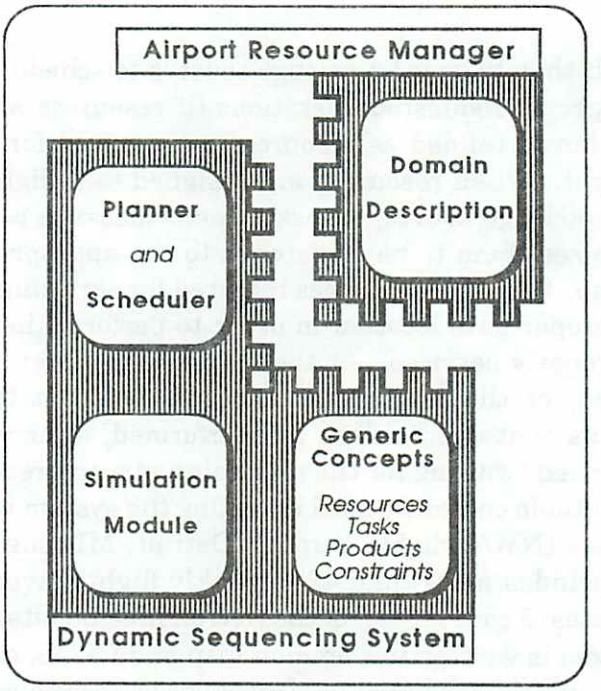

Figure 1: Dynamic Sequencing System Architecture

processes required for their production. A task network is a directed graph structure that describes the potential sequences of operations that contribute to the production of a product. Nodes in the task network represent either operations or logical connectors. Arcs between the nodes represent dependency and temporal sequencing constraints among the nodes. Task networks are instantiated by the planner and refined by the scheduler. An example of a typical task network for a product is presented in Figure 2, and is used in the airline resource management domain described in section 4. Our system provides the means for defining and linking together orders, products and task networks.

Operation nodes are defined by ovals that contain the name of the operation, the resource(s) required to perform the operation, and spaces for storing the start and end times of a window for processing the operation as well as the expected processing time. The time slots are filled in as part of the task network refinement. Operation nodes enclosed by non-filled rectangles represent operations whose requirement of resources is dependent on various conditions (like aircraft model or size). Operation nodes enclosed by filled rectangles represent operations that always require resources. Operation nodes that are not enclosed represent operations that do not require resources at all. The directed arcs between the nodes represent sequencing constraints between the operations. An operation cannot begin processing until all operations preceding it have been completed.

Figure 2 does not show all of the information provided in the standard task network. One of the important aspects that is missing is the dependency relation between the work site operation (which requires the securing of a gate resource), and all other servicing operations whose required resources are vehicles that must report to a particular work site. The work site must be secured before these other resources, because the assignment of these resources requires knowledge of the location where

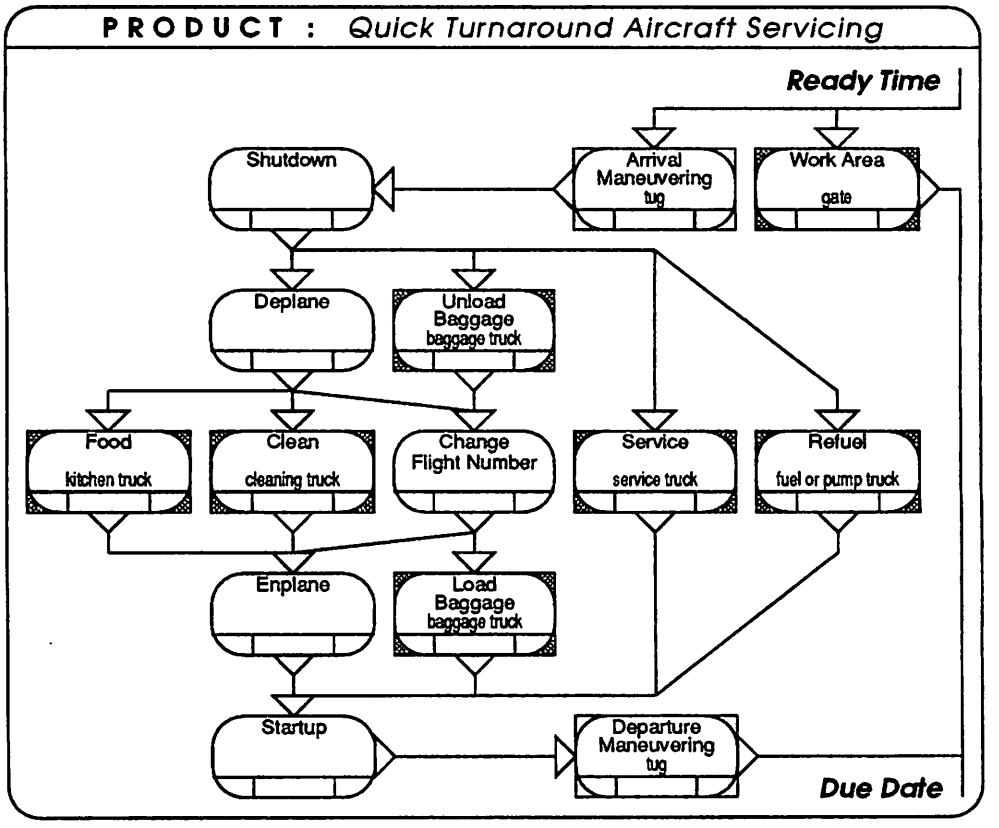

Figure 2: Simplified Task Network

they will be used.

Some task networks include logical connector nodes in order to allow for the representation of a variety of potential operation sequences for a single order. For example, if two different sequences of operations can both satisfy an order, then the two can be linked with a logical *or* node that tells the planner and scheduler to focus on just one of the two paths.

6.2 The Planner and Scheduler

Requests for service are presented as orders to the system. Depending on the nature of the domain, a schedule of anticipated orders may or may not be available. Domain information (including due dates and priorities) is used to rate the order. When an order is selected for planning, domain information provides the planner with the skeletal task network corresponding to the requested product. Skeletal task networks are refined in a least-commitment fashion by securing resources for the operation nodes in the network and propagating the implications of the resource assignments throughout the network.

Operation nodes may or may not require that resources be scheduled for them. If not, they are essentially ignored by the planner and scheduler. Operation nodes that do require resource scheduling trigger the scheduler when they become active. Operation nodes become active when the conditions provided in their corresponding operation definitions are satisfied (generally as the result of having all their preceding

and succeeding operations completely specified). Resources are secured by a generic knowledge source that attempts to find an available resource of the desired type for the requested time, considering the possible traveling and set-up times, and a particular relaxation method. If an attempt to secure a resource for an operation node fails, then further relaxation methods for the operation (specified in the domain knowledge base) are tried, until success is achieved or the relaxation methods are exhausted. Once a resource has been scheduled for an operation node, the effects of the assignment are propagated throughout the network, allowing other task nodes to become active in the process, and perhaps generating conflicts between nodes.

As the task network is refined, the system will periodically send commands to the outside world (simulator) so that required resources can be dispatched and prepared for their assigned jobs. Once a complete path of operation nodes extending from the ready time to the due date has been planned and all necessary resources have been secured, the task network represents a completed process plan ready for execution.

Each operation node in a task network maintains a window of time during which the operation for which it is responsible can be performed without disturbing the timings of other activities specified in the network. The slack provided in the time window, combined with information on the availability of resources, provides a metric for determining how much difficulty can be expected in attempting to secure a resource for the desired period of time within the window. As the size of this window is increased and decreased due to loosening and tightening (respectively) of other constraints in the network, and as the availability of resources within the environment changes, the difficulty of securing a resource for the specified time slot also changes. If the desired resource is not in adequate supply throughout the duration of the time window, then it is important to quickly attempt to secure that resource. In cases where the desired resource is readily available, the securing process can be (somewhat) delayed. In order to minimize the conflicts that may arise as the result of altering the time windows for activity nodes (some of which may already have secured their resources), our least-commitment strategy first schedules for highly-constrained nodes (those having time windows that intersect with periods of insufficient supply of their intended resources) so that fewer conflicts will be produced by changes to lesser-constrained nodes.

7 Work In Progress

While we have outlined the basic approach for a system to solve dynamic sequencing problems, there are certainly many research questions that have not yet been answered (let alone asked). In this section we discuss a number of these issues.

7.1 Representation of Resource Goals and Constraints

As mentioned earlier, the representation of the goals and constraints for resources has a great impact on the ability of the system to fully understand the viewpoints of all necessary objects when attempting to make decisions. In addition to orders, whose goals are stated through process plans (described by task networks) that indicate requests for resources and operation sequences to be performed, the resources must also be capable of specifying their own plans (scheduled work shifts and regular preventive maintenance sessions), and anticipating conflicts when their own plans

change.

For the moment, an approach is being considered that would make use of a task network of sorts for representing this knowledge. Resource managers responsible for each type of resource would schedule down time due to maintenance and shift changes by refining the process plans of their particular resources as these resources are assigned. One clear benefit would be the ability to use the same mechanism for representing the plans of both orders and resources, to allow the facilities for detecting constraint failures and conflicts to easily work on both entities. Experimentation with resource-based and order-based constraint conflicts will provide us with the necessary information to help make a proper decision, by emphasizing what kinds of conflicts occur, and what information is needed to successfully resolve them.

7.2 Constraint Relaxation

Constraint relaxation provides a means for coming up with a satisficing solution to a problem when a satisfying one cannot be produced. For example, if all but one operation has been set for a particular job, and the resource required for the last operation is not available during the desired time, then the important questions are what constraints *can* and *should* be relaxed, and *how*? In ISIS, the most common form of relaxation is the generation of multiple search states to choose among while expanding the solution graph using a beam search. In our system, since a least-commitment approach is used, we want to rethink our decisions (backtrack) only when it is absolutely necessary. So, we need to know how to represent the potential relaxations for a particular situation, and we need to know when, and in what order to perform them.

One potential approach to the problem of relaxing resource constraints on operations is to specify for operations a number of alternate actions that can be taken (and perhaps their repercussions) in order to secure resources if previous methods for obtaining the same resource fail. The scheduler would thus have a variety of methods to use to secure resources, and would even be able to retry earlier failed methods when certain changes occur in the environment. This approach, however, only addresses the resource allocation process. There are other constraints in the environment. It will be interesting to see if a single generic approach can address the relaxation process for all kinds of constraints, or whether different approaches for different kinds of constraints will be required.

7.3 Constraint Failure and Conflict Resolution

The process of dealing with constraint failure and conflict resolution includes first the detection and localization of the problem, then the analysis of the situation, and finally the formulation of a resolution strategy. The detection and localization of the problem is made easier by the task network and the representation of constraints. The difficulties arise with the remaining steps. Analysis of the situation requires a broad understanding of the objects and constraints involved. Formulating an adequate response to the conflict, however, requires knowledge of the relaxation methods available, an ability to project the results of potential actions, and finally an understanding of the system's present processing goals, as in whether time considerations warrant immediate action or postponement of the response.

As with these other open questions, extensive experimentation will be required to provide us with sample cases to use in gaining a better understanding of all that is going on here so that a clear consistent approach can be designed and implemented.

7.4 Information and Control

The complete description of a dynamic sequencing problem domain requires information about resources, operations, and orders, as well as a variety of constraints on how they interact and the dependencies between them. It must also be determined how this information will be represented and used by the components in the system for the purpose of planning and scheduling task and resource sequences for orders. In addition, the control of the system depends on the manipulation of various kinds of information in order to intelligently decide on the courses of action to take in different situations. We have a general idea of the kinds of information required, and how to represent it so that it can be exploited by the various components of the system, and we have a general idea of how the control of the system will be organized, but it is not clear how it will all interact. Much of what we have to learn about the information and control in dynamic sequencing problems will become clearer in the process of implementing the system and exploring its behavior in (simulated) real-world situations.

8 Summary

In this paper we have described the important class of real-world dynamic sequencing problems and presented the design for an integrated, knowledge-based system for solving them. The major features of this design are the integration of multiple scheduling perspectives and the use of an opportunistic, least-commitment problem-solving approach that considers all important viewpoints in making its decisions, reacts quickly to unpredictable and changing environmental conditions, and minimizes the need to backtrack as the result of hasty, uninformed planning and scheduling decisions. Such an integrated approach is needed to solve real-world dynamic sequencing problems.

References

[Bellman *et al.*, 1982] R. Bellman, A.O. Esogbue, and I. Nabeshima. *Mathematical Aspects of Scheduling and Applications*. Pergamon Press, Oxford, 1982.

[Blackboard Technology Group, Inc., 1990] Blackboard Technology Group, Inc. *GBB Reference: Version 2.0*. Amherst, Massachusetts, 1990.

[Brazile and Swigger, 1988] R.P. Brazile and K.M. Swigger. GATES: An airline gate assignment and tracking expert system. *IEEE Expert*, 3(2):33–39, Summer 1988.

[Coffman, 1976] E.G. Coffman, Jr., editor. *Computer and Job-Shop Scheduling Theory*. John Wiley & Sons, New York, 1976.

[Fox and Smith, 1984] M.S. Fox and S.F. Smith. ISIS: A knowledge-based system for factory scheduling. *Expert Systems*, 1:25–49, 1984.

[Fox *et al.*, 1982] M.S. Fox, B.P. Allen, and G.A. Strohm. Job shop scheduling: An investigation in constraint-directed reasoning. In *Proceedings of the Second National Conference on Artificial Intelligence*, pages 155–158, 1982.

[Fox, 1983] M.S. Fox. *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*. PhD thesis, Computer Science Dept., Carnegie Mellon Univ., 1983.

[French, 1982] S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Ellis Horwood Limited, Chichester, 1982.

[Ow and Smith, 1986] P.S. Ow and S.F. Smith. Toward an opportunistic scheduling system. In *Proceedings of the Nineteenth Hawaiian International Conference on System Sciences*, pages 345–353, 1986.

[Ow *et al.*, 1988] P.S. Ow, S.F. Smith, and A. Thiriez. Reactive plan revision. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 77–82, 1988.

[Pease, 1978] M.C. Pease, III. ACS.1: An experimental automated command support system. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(10):725–735, October 1978.

[Smith and Ow, 1985] S.F. Smith and P.S. Ow. The use of multiple problem decompositions in time constrained planning tasks. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 1013–1015, 1985.

[Smith *et al.*, 1986a] S.F. Smith, M.S. Fox, and P.S. Ow. Constructing and maintaining detailed production plans: Investigations into the development of knowledge-based factory scheduling systems. *AI Magazine*, 7(4):45–61, Fall 1986.

[Smith *et al.*, 1986b] S.F. Smith, P.S. Ow, C. LePape, B. McLaren, and N. Muscettola. Integrating multiple scheduling perspectives to generate detailed production plans. In *Proceedings of the SME Conference on Artificial Intelligence in Manufacturing*, 1986.