MICRO-OPPORTUNISTIC SCHEDULING:
THE MICRO-BOSS FACTORY SCHEDULER

Norman Sadeh

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

This research was supported, in part, by the Defense Advanced Research Projects Agency under contract #F30602-91-F-0016 and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

1 Introduction

In a global market economy, the need for cost-efficient production management techniques is becoming more critical every day. In contrast with this need, current production management practice is too often characterized by low levels of due date satisfaction, high levels of inventory and, more generally, a state of chaos, in which the computer systems that are used to provide managerial guidance do not accurately reflect the current state of affairs, as they rely on oversimplified and rigid models of the production environment. A major challenge for research in this area is to develop new production management techniques and tools that (1) can account more precisely for actual production management constraints and objectives, (2) are better suited for handling production contingencies, and (3) allow the user to interactively manipulate the production schedule to reflect idiosyncratic constraints and preferences not easily amenable to representation in the computer model. This paper describes Micro-Boss, a decision-support system for factory scheduling currently under development at Carnegie Mellon University. Micro-Boss aims at generating and maintaining high-quality realistic production schedules by combining powerful predictive, reactive and interactive scheduling capabilities. To this end, the system relies on so-called *micro-opportunistic* search techniques that enable the scheduling system to constantly revise its scheduling strategy during the construction or repair of a schedule. These techniques are shown to be more effective than less flexible scheduling techniques proposed in the Operations Research and Artificial Intelligence literature.

1.1 The Production Scheduling Problem

Production scheduling requires allocating resources (e.g. machines, tools, human operators) over time to a set of jobs while attending to a variety of constraints and objectives.

Typical constraints include:

- *functional constraints* limiting the types of operations that a specific resource can perform,

- *capacity constraints* restricting the number of jobs a resource can process at once,

- *availability constraints* specifying when each resource is available (e.g. number of shifts available on a group of machines),

- *precedence constraints* between the operations in a job, as specified in the job's process routing,

- *processing time constraints* specifying how long it usually takes to perform each operation,

- *setup constraints* requiring that each machine be in the proper configuration before performing a particular task (e.g. proper sets of fixtures and tools),

- *time-bound constraints*: Each job typically has an earliest acceptable release date before which it cannot start (e.g. because its raw materials cannot arrive earlier) and a due date by which ideally it should be delivered to a customer.

Some of these constraints must be satisfied for a schedule to be valid (so-called non-relaxable or hard constraints). For instance, milling operations can only be performed on milling machines. Other groups of constraints are not always satisfiable and may need to be relaxed (so-called relaxable or soft constraints). For instance, due date constraints often need to be relaxed for a couple of jobs due to the limited capacity of the production facility. Availability constraints are another example of constraints that can be relaxed, by either working overtime or adding extra shifts. A good schedule is one that satisfies all hard constraints while selectively relaxing soft constraints so as to help the company maximize profits.

Two factors that critically influence the quality of a schedule are due date satisfaction and inventory levels. Missing a customer due date may result in tardiness penalties, loss of customer orders, delayed revenue receipts, etc. Inventory costs include interests on the costs of raw materials, direct inventory holding costs, interests on processing costs, etc. One often distinguishes between in-process inventory costs (also referred to as work-in-process inventory costs) and finished-goods inventory costs. Work-In-Process (WIP) inventory costs account for inventory costs resulting from orders that have not yet been completed, while finished-goods inventory costs result from completed orders that have not yet been shipped to customers.

Manufacturing contingencies such as machine breakdowns, late arrivals of raw materials, variations in operation durations and yields further complicate production scheduling. In the face of contingencies, schedules need to be updated to reflect the new state of affairs. The sheer size of most factory scheduling problems precludes the generation of new schedules from scratch each time an unanticipated event occurs. In fact, most contingencies do not warrant such extreme actions and are best handled by repairing a portion of the existing schedule [3].

As schedules are optimized at a more detailed level, they may also become more sensitive to disruptions and require more frequent repairs. In general, there is a limit to the amount and detail of information that one can reasonably expect to represent in a computer model. For instance, a worker's preference for performing more demanding tasks in the morning may not be worth storing in the computer model and, instead, may be best accounted for by allowing the end-user to interactively manipulate the schedule.

Even under idealized conditions such as simplified objectives (e.g. minimizing total tardiness or maximizing throughput) and deterministic assumptions, scheduling has been shown to be an NP-hard problem [12, 14, 11]. Uncertainty further adds to the difficulty of the problem, and makes it even more impractical to look for optimal solutions. Instead, practical approaches to production scheduling are heuristic in nature. The next subsection briefly reviews earlier approaches to production scheduling, identifies some of their shortcomings, and introduces a new search paradigm, called micro-opportunistic search, that shows promise for addressing some of these shortcomings.

1.2 A Micro-opportunistic Approach to Production Scheduling

To this date, the most widely used computer-based approach to production scheduling remains by far the Material Requirements Planning (MRP) or Manufacturing Resource Planning (MRP-II) approach developed in the seventies [22, 40, 41]. In this approach, demand for end-products as specified in a Master Production Schedule is exploded into time-phased requirements for component items (subassemblies, parts, raw materials, etc.) required for the production of these end-products[1]. Because their time-phasing logic relies on standard operation leadtimes that do not account for the actual load of the production facility, MRP systems often fail to produce realistic schedules. They sometimes overload the facility, thereby causing orders to be delivered late. In an attempt to alleviate this problem, MRP systems often pad the schedule by inserting generous "safety" leadtimes. These safety leadtimes tend to be rather arbitrary and produce unnecessarily large amounts of inventory. In fact, because they are often unrealistic and are not meant to be updated in real-time[2], MRP schedules are not directly used to schedule production but rather to assign priorities to jobs [27, 39]. These priorities in turn determine the order in which jobs are actually processed at each work center.

Shortcomings of the traditional MRP approach reflect limitations of computing technologies available in the seventies. In the eighties with the advent of more powerful computers, several more sophisticated techniques emerged [13, 9, 23, 1, 24, 21]. The first and by far most publicized of these techniques is the one developed by Goldratt and his colleagues within the context of the OPT factory scheduling system [13, 15, 10]. OPT demonstrated the benefits of building detailed production schedules that account for the actual load of the plant and the finite capacity of its resources ("finite scheduling" approaches). This system also underscored the potential benefits of distinguishing between bottleneck and non-bottleneck resources [15, 10]. In OPT, bottlenecks are scheduled first to optimize the throughput of the plant. Later, the production schedule is completed by compactly scheduling non-bottleneck operations so as to reduce inventory. The distinction between bottleneck and non-bottleneck machines was pushed one step further in the OPIS system [34, 24], as it was recognized that new bottlenecks can appear during the construction of the schedule. The OPIS scheduler combines two scheduling perspectives: a resource-centered perspective for scheduling bottleneck resources, and a job-centered perspective to schedule non-bottleneck operations on a job-by-job basis. Rather than relying on its initial bottleneck analysis, OPIS typically repeats this analysis each time a resource or a job has been scheduled. This ability to detect the emergence of new bottlenecks during the construction of the schedule and revise the current scheduling strategy has been termed

[1]For instance, if an end-product required by the end of week 2 is obtained by assembling two sub-components and the assembly process typically takes a week to be completed, both sub-components will be required by the end of week 1.

[2]MRP systems are generally run on a weekly, possibly even a monthly basis.

opportunistic scheduling [24]. Nevertheless, the opportunism in this approach remains limited in the sense that it typically requires scheduling an entire bottleneck (or at least a large chunk of it) before being able to switch to another one. For this reason, we actually refer to these techniques as *macro-opportunistic*.

In fact, variations in the job mix over time often cause different machines (or groups of machines) to be bottlenecks over different time intervals. Bottlenecks are sometimes said to "wander over time". Also, as a schedule is constructed for a bottleneck machine, a new machine may become more constraining than the original bottleneck. For instance, scheduling decisions on a bottleneck machine may require that a large number of jobs be processed on a preceding machine over a short period of time. At some point during the construction of the schedule, contention for the preceding machine may become higher than that for the original bottleneck. A scheduling technique that can only schedule large resource/job subproblems will not be able to take such considerations into account. It will overconstrain its set of alternatives before having worked on the subproblems that will most critically affect the quality of the entire schedule. This in turn will often result in poorer solutions. A more flexible approach would stop scheduling operations on a resource as soon as another resource is identified as being more constraining. In the presence of multiple bottlenecks, such a technique would be able to shift attention from one bottleneck to another during the construction of the schedule rather than focus on the optimization of a single bottleneck at the expense of others. This paper presents such a flexible approach to scheduling. We call it *micro-opportunistic* scheduling. In this approach, resource contention is continuously monitored during the construction of the schedule, and the problem solving effort constantly redirected towards the most serious bottleneck resource. In its simplest form, this micro-opportunistic approach results in an *operation-centered* view of scheduling, in which each operation is considered an independent decision point and can be scheduled without requiring that other operations using the same resource or belonging to the same job be scheduled at the same time[3].

---

[3]An alternative approach in which resources can be resequenced to adjust for resource schedules built further down the road is described in [1] and [7]. This approach has been very successful at minimizing makespan, namely the total duration of the schedule. This measure is closely related to the throughput of the plant but does not account for individual job due dates, individual job tardiness costs or inventory costs. Attempts to generalize the procedure to account for due dates seem to have been less successful so far [33]. It should be pointed out that the idea of continuously reoptimizing the current partial schedule is compatible with a micro-opportunistic approach.

Experimental results presented at the end of this paper indicate that micro-opportunistic scheduling procedures often yield better schedules than less flexible bottleneck-centered approaches. Because of their flexibility, micro-opportunistic scheduling heuristics also seem particularly well suited to solving problems in which some operations have to be performed within non-relaxable time windows [29, 31] as well as repairing schedules in the face of contingencies. Finally, we find that they can easily be integrated in interactive systems in which manual and automatic scheduling decisions can be interleaved, thereby allowing the user to incrementally manipulate and compare alternative schedules (e.g. "What-if" type of analysis).

1.3 Paper Outline

The remainder of this paper successively reviews the predictive, reactive and interactive capabilities of the Micro-Boss scheduling system.

Section 2 describes the micro-opportunistic search procedure implemented in Micro-Boss, focusing on look-ahead techniques used to measure contention, and heuristics to identify and schedule critical operations. A small example illustrating the use of these techniques is provided in Section 3. Section 4 describes the reactive and interactive components of the system. Section 5 reports the results of an experimental study comparing Micro-Boss against several popular scheduling approaches, including a coarser opportunistic scheduler, under a wide range of simulated situations. Finally, Section 6 briefly reviews current research efforts and summarizes the impact of this work.

2 A Micro-opportunistic Search Procedure

In this section, a deterministic scheduling model is assumed, in which all jobs to be scheduled are known in advance. Issues pertaining to reactive scheduling and control in the face of manufacturing contingencies such as machine breakdowns are addressed in a later section.

2.1 A Deterministic Scheduling Model

For the time being, we consider a deterministic scheduling problem in which a set of jobs $J=\{j_1,...,j_n\}$ has to be scheduled on a set of physical resources $RES=\{R_1,...,R_m\}$. Each job j_l consists of a set of operations $O^l=\{O^l_1,...,O^l_{n_l}\}$ to be scheduled according to a process routing that specifies a partial ordering among these operations (e.g. O^l_i BEFORE O^l_j). We further assume scheduling problems with in-tree process routings, namely process routings in which operations can have one or several direct predecessors but at most one direct successor (e.g. assembly process routings). This is by far the most common type of process routing encountered in manufacturing.

Additionally, each job j_l has an earliest acceptable release date, erd_l, a due-date, dd_l, and a latest acceptable completion date, lcd_l, where $lcd_l \geq dd_l \geq erd_l$. All jobs need to be scheduled between their earliest acceptable release date and latest acceptable completion date[4]. The earliest acceptable release date may correspond to the earliest possible arrival date of raw materials. It is assumed that the actual release date (or job start date) will be determined by the schedule that is constructed. The latest acceptable completion date may correspond to a date after which the customer will refuse delivery. If such a date does not actually exist, it can always be chosen far enough in the future so that it is no longer a constraint.

Each operation O^l_i has an expected duration, du^l_i, and a start time, st^l_i (to be determined), whose domain of possible values is delimited by an earliest start time, est^l_i, and a latest start time, lst^l_i (initially derived from the job's earliest acceptable release date erd_l and latest acceptable completion date lcd_l). We assume that each operation O^l_i requires a single resource R^l_i[5] The model further allows for resource availability constraints that specify the times when each resource is normally available (e.g. number of shifts and whether the resource is available over

---

[4]Notice that this formulation does not exclude infeasible problems.

[5]Work is currently under way to allow for parallel machines, including machines with different processing speeds.

the week-end). Finally, setup operations may be required before an operation can start on a machine. Examples of setup operations include changing the fixtures holding a part, loading a new part, cleaning a painting station when switching from one color to another, etc.

The objective of the scheduling system, under deterministic assumptions, is to build a schedule that satisfies the above constraints while minimizing (as much as possible) the costs incurred for missing due dates or carrying overhead inventories. These costs are briefly described below.

<u>COSTS</u>:

Each job j_l has:

• a **marginal tardiness cost**, $tard_l$: the cost incurred for each unit of time that the job is tardy (i.e. finishes past its due date). Marginal tardiness costs generally include tardiness penalties, interests on delayed profits, loss of customer goodwill, etc[6]. The tardiness cost of job j_l, in a given schedule, is:

$$TARD_l = tard_l \times Max(0, C_l - dd_l) \tag{1}$$

where $C_l = st^l_{n_l} + du^l_{n_l}$ is the completion date of job j_l in that schedule, assuming that $O^l_{n_l}$ is the last operation in job j_l.

• **marginal in-process and finished-goods inventory costs**: In our model, each operation O^l_i can incrementally introduce its own non-negative marginal inventory cost, inv^l_i. Typically the first operation in a job introduces marginal inventory costs that correspond to interests on the costs of raw materials, interests on processing costs (for that first operation), and marginal holding costs. Downstream operations[7] introduce additional marginal inventory costs such as interests on processing costs or interests on the costs of additional raw materials required by these operations. The total inventory cost for a job j_l, in a given schedule, is:

$$INV_l = \sum_{i=1}^{n_l} inv^l_i \times [Max(C_l, dd_l) - st^l_i] \tag{2}$$

The total cost of a schedule is obtained by summing the cost of each job schedule:

---

[6]In this model, inventory costs incurred after the due date are not included in the tardiness costs but rather in the inventory costs described below.

[7]An operation O^k_i is said to be downstream (upstream) of another operation O^k_j within the same job if O^k_i is a direct or indirect successor (predecessor) of O^k_j in that job, as defined by its process routing.

$$\boxed{\mathit{Schedule\ Cost} = \sum_{l=1}^{n} (TARD_l + INV_l)} \qquad (3)$$

A SMALL EXAMPLE:

Figure 1 depicts a small scheduling problem with 4 jobs that will be used in this section to illustrate the behavior of the micro-opportunistic scheduling heuristics implemented in Micro-Boss. Each square box represents an operation and is labeled by the name of this operation (e.g. O_1^1), its (expected) duration (e.g. $du_1^1 = 2$), and its resource requirement (e.g. $R_1^1 = R_1$). In this simple example, each operation is assumed to require a single resource, for which there are no substitutes. The arrows represent precedence constraints. For instance, job j_1 requires 5 operations $O_1^1, O_2^1, ..., O_5^1$. O_1^1 has to be performed before O_2^1, O_2^1 before O_4^1, etc. The other arcs in the graph represent capacity constraints which require that each resource be allocated to only one operation at a time. There is a capacity constraint between each pair of operations that require the same resource. Notice that R_2 is the only resource required by four operations (one from each job). Notice also that, in three out of four jobs (namely j_1, j_3, and j_4), the operation requiring R_2 is one of the job's longest operations. Consequently, resource R_2 can be expected to be the main bottleneck of the problem. We will see that, to some extent, resource R_1 constitutes a secondary bottleneck.

Figure 1 should be about here.

The earliest acceptable release dates, due dates, and latest acceptable completion dates of the jobs are provided in Table 1 along with the marginal tardiness and inventory costs of these jobs.

Table 1 should be about here.

2.2 Overview of the Search Procedure

In Micro-Boss, each operation is considered an independent decision point. Any operation can be scheduled at any time, if deemed appropriate by the system. There is no obligation to simultaneously schedule other operations upstream or downstream within the same job, nor is there any obligation to schedule other operations competing for the same resource.

Micro-Boss proceeds by iteratively selecting an operation to be scheduled and a reservation (i.e. a start time and a set of resources to fulfill the resource requirements of the operation) to be assigned to that operation. Every time an operation is scheduled, a new *search state* is created, where new constraints are added to account for the reservation assigned to that operation. A consistency enforcing procedure then takes care of updating the set of remaining possible reservations of each unscheduled operation. If an unscheduled operation is found to have no possible reservations left, a *deadend state* has been reached, in which case the system needs to *backtrack* (i.e. it needs to undo some earlier reservation assignments to be able to complete the schedule). If the search state does not appear to be a deadend, the system moves on and looks for a new operation to schedule and a reservation to assign to that operation.

In Micro-Boss, search efficiency[8] and schedule quality are maintained at a high level by interleaving search with the application of consistency enforcing techniques and a set of look-ahead techniques that help decide which operation to schedule next (*operation ordering heuristic*) and which reservation to assign to that operation (*reservation ordering heuristic*).

1. **Consistency Enforcing/Checking:** Consistency enforcing techniques prune the search space by inferring new constraints resulting from earlier reservation assignments [19, 30]. By constantly accounting for earlier scheduling decisions, these techniques reduce the chances of reaching a deadend (i.e. a partial schedule that cannot be completed without backtracking). Simultaneously, by allowing for the early detection of deadend states, these techniques limit the amount of work wasted in the exploration of fruitless alternatives.

2. **Look-ahead Analysis**: A two-step look-ahead procedure is applied in each search state, which first optimizes reservation assignments within each job, and then, for each resource, computes contention between jobs over time. Resource/time intervals where job contention is the highest help identify the critical operation to be scheduled next (*operation ordering heuristic*). Reservations for that operation are then ranked according to their ability to minimize the costs incurred by the conflicting jobs (*reservation ordering heuristic*). By constantly redirecting its

[8]We define search efficiency as the ratio of the number of operations to be scheduled over the number of search states generated. If the number of search states generated to build the schedule is equal to the number of operations, search efficiency is equal to 1.

effort towards the most serious conflicts, the system is able to build schedules that are closer to the global optimum. Simultaneously, because the scheduling strategy is aimed at reducing job contention as rapidly as possible, chances of reaching deadend states tend to quickly subside too.

The opportunism in Micro-Boss results from the ability of the system to constantly *revise its search strategy and redirect its effort towards the scheduling of the operation that appears to be the most critical in the current search state*. This degree of opportunism differs from the one displayed by earlier approaches where scheduling entities were large resource/job subproblems [24, 6], i.e. where large resource/job subproblems had to be scheduled before the system could revise its scheduling strategy.

Concretely, given a scheduling problem such as the one described in Figure 1, Micro-Boss starts in a search state in which no operation has been scheduled yet[9], and proceeds according to the following steps:

1. If all operations have been scheduled then stop, else go on to 2;
2. Apply the **consistency enforcing** procedure;
3. If a deadend is detected then **backtrack**, else go on to step 4;
4. Perform a **look-ahead** analysis: rank the possible reservations of each unscheduled operation according to how well they minimize the costs of the job to which the operation belongs (step 1), and evaluate job contention over time for each resource (step 2);
5. Select the next operation to be scheduled (i.e. **operation ordering** heuristic);
6. Select a reservation for that operation (i.e. **reservation ordering** heuristic)
7. Create a **new search state** by adding the new reservation assignment to the current

[9]Alternatively, Micro-Boss can also complete a partial schedule, in which case the initial search state corresponds to the initial partial schedule. A description of reactive and interactive capabilities of the system is provided in Section 4.

partial schedule. Go back to 1.

As in other constraint-directed scheduling systems [16], the consistency enforcing procedure used in Micro-Boss (1) maintains for each unscheduled operation a pair of earliest/latest possible start times and (2) marks as unavailable those resource/time intervals allocated to already scheduled operations. Additionally, reservation pruning performed by the Micro-Boss consistency procedure also accounts for resource/time intervals that are absolutely needed by unscheduled operations. Figure 2 displays an example of an unscheduled operation O_i^k whose earliest and latest possible reservations overlap. Whichever reservation this operation is ultimately assigned, it will always need time interval $[lst_i^k, eft_i^k]$. Accordingly, the Micro-Boss consistency procedure prunes the set of remaining possible reservations of other unscheduled operations requiring that resource by removing all those reservations that overlap with time interval $[lst_i^k, eft_i^k]$[10].

Figure 2 should be about here.

Results presented in this paper were obtained using a simple chronological backtracking scheme. Experimentation with more sophisticated backtracking schemes are described in [31].

The remainder of this section gives a more detailed description of the look-ahead analysis and the operation/reservation ordering heuristics used in Micro-Boss. Further details on these techniques as well as other aspects of the system can be found in [30].

2.3 Look-ahead Analysis in Micro-Boss

2.3.1 Optimizing Critical Conflicts First

If all jobs could be scheduled optimally (i.e. just-in-time), there would be no scheduling problem. Generally, this is not the case. Jobs typically have conflicting resource requirements. The look-ahead analysis carried out by Micro-Boss in each search state aims at helping the scheduling system focus its effort on those conflicts that currently appear most critical. A critical conflict is one that will require an important tradeoff, i.e. a tradeoff that will significantly

[10]This differs from an earlier version of the system [30], in which resource/time intervals needed by unscheduled operations were only used to detect conflicts. In this earlier version, a conflict would be detected when two or more unscheduled operations needed overlapping resource/time intervals. Rather than waiting for such conflicts to arise, our new consistency procedure efficiently prevents such conflicts from occurring, thereby further reducing backtracking. A generalized version of this procedure is used for parallel machines.

impact the quality of the *entire* schedule. By first focusing on critical conflicts, Micro-Boss ensures that it has as many options as possible to optimize these conflicts. As illustrated by a trace provided in the next section, once critical tradeoffs have been worked out, the remaining unscheduled operations tend to become more decoupled and hence easier to optimize[11]. As contention subsides, so does the chance of needing to backtrack. In other words, by constantly redirecting search towards those tradeoffs that appear most critical, Micro-Boss is expected to produce better schedules while simultaneously keeping backtracking at a low level.

More specifically, a two-step look-ahead procedure is applied to each search state. This procedure first optimizes reservation assignments within each job, and then, for each resource, computes contention between jobs over time. The so-called *demand profiles* produced by these computations help identify operations whose good reservations (as identified in the first step) conflict with the good reservations of other operations. These operations define the critical conflicts on which Micro-Boss works first.

This two-step look-ahead analysis is further detailed below.

2.3.2 Step 1: Reservation Optimization Within a Job

In order to detect critical conflicts between the resource requirements of unscheduled operations, Micro-Boss keeps track of the best start times that remain available to each unscheduled operation within its job. Additionally the system implicitly maintains for each remaining possible start time τ of each unscheduled operation O_i^k, a function $mincost_i^k(\tau)$, that indicates the minimum additional costs that would be incurred by job j_k (the job to which O_i^k belongs), if O_i^k was to start at $st_i^k = \tau$ rather than at one of its best possible start times. By definition, if $st_i^k = \tau$ is one of the best start times that remain available to O_i^k within its job, then $mincost_i^k(\tau) = 0$. Rather than explicitly maintaining *mincost* functions, Micro-Boss simply maintains for each unscheduled operation (1) an apparent marginal tardiness cost incurred by the job for each unit of time that the operation starts past its latest best start time and (2) an apparent marginal inventory cost for each unit of time that the operation starts before its earliest best start time. These costs are updated in each search state to account for earlier scheduling decisions, using a set of efficient propagation procedures described in [30].

[11]This is similar to the way bottleneck schedules drive other scheduling decisions in OPT.

2.3.3 Step 2: Building Demand Profiles to Identify Highly Contended Resource/Time Intervals

In Micro-Boss, critical conflicts are identified as groups of operations whose good reservations (within their jobs) conflict with each other. The importance of a conflict depends on the number of jobs that are competing for the same resource, the amount of temporal overlap between the requirements of these jobs, the number of alternative reservations still available to the conflicting operations and the differences in cost between these alternative reservations (as determined by the *mincost* functions computed in step 1).

To identify critical conflicts, Micro-Boss uses a probabilistic framework, in which each remaining possible start time τ of an unscheduled operation O_i^l is assigned a *subjective probability* $\sigma_i^l(\tau)$ to be selected for that operation (in the final schedule). Possible start times with lower *mincost* values are assigned a larger probability, thereby reflecting our expectation that they will yield better schedules. Given these start time probability distributions, the probability that an unscheduled operation O_i^l uses its resource at time t, which is referred to as the *individual demand* of O_i^l for R_i^l, is:

$$D_i^l(t) = \sum_{t - du_i^l < \tau \leq t} \sigma_i^l(\tau) \tag{4}$$

where du_i^l is the duration of O_i^l. $D_i^l(t)$ is also a (subjective) measure of the reliance of operation O_i^l on the availability of its resource at time t. By adding the individual demands of all unscheduled operations requiring a given resource, say R_k, the system obtains an *aggregate demand profile*, $D_{R_k}^{aggr}(t)$, that indicates contention between (all) unscheduled operations for that resource R_k as a function of time:

$$D_{R_k}^{aggr}(t) = \sum D_i^l(t) \tag{5}$$

where the summation is carried over all unscheduled operations that need resource R_k.

Figure 3 should be about here.

Figure 3 displays $\sigma_2^2(\tau)$, the start time distribution of operation O_2^2 in the problem defined in Figure 1. This start time distribution is depicted in the initial search state, where all operations still have to be scheduled. In this search state, start time $st_2^2 = 9$ is the best possible start time for O_2^2: it corresponds to a just-in-time schedule of job j_2. Later start times have a lower subjective probability as they would force the job to finish after its due date. Earlier start times are also suboptimal since they would produce additional inventory. In this example, the marginal tardiness cost of job j_2, $tard_2 = 20$, is four times larger than the marginal inventory cost introduced by operation O_1^2, $inv_1^2 = 5$. Accordingly $\sigma_2^2(\tau)$ has a steeper slope for $\tau > 9$ than for $\tau < 9$. Additional details on how these distributions are constructed can be found in [30].

Figure 4 should be about here.

Figure 4 displays the individual demand profiles of the four operations requiring resource R_2. These demand profiles represent the subjective probability that each one of these operations uses resource R_2 as a function of time. The aggregate demand for resource R_2 is obtained by summing these four individual demands over time. The individual demands of operations O_3^3 and O_2^4 are quite uniform since these two operations have relatively low apparent marginal costs (See the marginal tardiness and inventory costs of job j_3 and job j_4 in Table 1). In contrast, operations O_2^1 and O_2^2, which have larger apparent marginal costs, have individual demands that are concentrated around their best reservations.

Figure 5 should be about here.

Similar computations can be performed for each of the five resources in the problem. The resulting aggregate demands (in the initial search state) are displayed in Figure 5. As expected, resource R_2 appears to be the most contended for. The aggregate demand for that resource is well above 1.0 over a large time interval, with a peak at 1.49. Resource R_1 appears to be a potential bottleneck at the beginning of the problem, with a demand peaking at 1.20. Whether R_1 will actually be an auxiliary bottleneck or not cannot be directly determined from the curves displayed in Figure 5. Instead the system needs to update these curves in each search state to account for earlier decisions. It could be the case that, as operations requiring R_2 are scheduled, the aggregate demand for R_1 becomes smoother. In this example, this is not the case. On the contrary, after only a portion of the operations requiring resource R_2 have been scheduled, Micro-Boss will redirect its effort towards the scheduling of resource R_1, as indicated by the trace provided in Section 4.

2.4 Operation Selection

Critical operations are identified as operations whose good reservations (as identified in the first step of the look-ahead analysis) conflict with the good reservations of other operations. The largest peak in the aggregate demand profiles determines the next conflict (or micro-bottleneck) to be optimized; the operation with the largest reliance on the availability of the corresponding resource/time interval (i.e. the operation with the largest individual contribution to the peak) is selected to be scheduled next. Indeed, this operation is the one whose good reservations are the most likely to become unavailable if other operations contending for the current micro-bottleneck were scheduled first.

Figure 6 should be about here.

In the example introduced earlier, the most contended demand peak is the one for resource R_2 over interval [7,12[. Figure 6 displays the aggregate demand for resource R_2 together with the individual demands of the four operations requiring this resource. The operation with the largest contribution to the demand peak is O_2^1. Therefore this operation is selected to be scheduled next. This is no real surprise: O_2^1 belongs to one of the two jobs in the problem that have a high marginal tardiness cost ($tard_1 = 20$). While any delay in starting job j_1 will cause this job to be late, job j_2 (i.e. the other job with a high marginal tardiness cost) can tolerate a small amount of delay without ending up late.

2.5 Reservation Selection

To assign a reservation to the critical operation selected to be scheduled next, Micro-Boss uses a hybrid reservation ordering heuristic that adapts to contention for the current critical resource/time interval.

When contention is high[12], the system attempts to identify a reservation (for the critical operation) that will reduce as much as possible the costs incurred by the job to which that operation belongs and by the other jobs with which that operation competes. This is approximated as a single-machine early/tardy scheduling problem in which operations scheduled past their best start times incur penalties determined by their apparent marginal tardiness costs, while operations scheduled before their best start times incur earliness penalties determined by their apparent marginal inventory costs (as determined by the *mincost* functions) [2, 30]. In the experiments reported at the end of this paper, several variations of a single-machine early/tardy procedure developed by Ow and Morton [26, 30] were successively run and the single-machine schedule with the lowest cost was used to determine the reservation assigned to the critical

[12]Currently contention is measured along two dimensions: (1) the average demand for the critical resource/time interval (δ) and (2) the contribution of the critical operation to the demand peak as a percentage of the total demand for that peak (χ). In the experiments presented in this paper, contention was considered high when both $\delta > 0.8$ and $\chi < 0.7$.

operation. More recently, a new scheduling heuristic has been developed to solve problems with setups [17]. We are currently in the process of extending this heuristic to also solve problems with parallel machines.

When contention is lower, the system dynamically switches to a "greedy" reservation ordering heuristic, in which reservations are simply rated according to their apparent costs (i.e. according to their *mincost* values). Indeed, in situations where contention is not too high, a sizable proportion of the good reservations of non-critical operations tend to remain available after more critical operations have been scheduled. When this is the case, a greedy reservation ordering heuristic is all that is needed.

3 A Small Example

Micro-Boss was originally developed in Knowledge Craft, a frame-based language that runs on top of Common Lisp and has recently been reimplemented in C++. The system runs on a DECstation 5000 under Mach UNIX. The small example used throughout this paper requires about 0.15 CPU seconds. An edited trace of this example is given in Figure 7.

Observe that, rather than entirely scheduling the main bottleneck resource, namely resource R_2, Micro-Boss started scheduling operations requiring resource R_1 after only two out of the four operations requiring R_2 had been scheduled. The average expected demand displayed in each search state is the average demand for the critical demand peak, and the average contribution is that of the critical operation for the demand over that peak. The decoupling effect of the operation ordering heuristic is very clear in this example. In particular, the average demand over the critical peak consistently decreases from one search state to the next, thereby indicating a regular decrease in contention as the schedule is constructed[13]. This observation is correlated by the average contribution of the critical operation to the demand peak in each search state. As the schedule is constructed, the contribution of the critical operation to the peak becomes a larger proportion of the total demand for that peak. This indicates that there are fewer and fewer operations contending with each other. After half of the operations have been scheduled (depth 7), contention has totally disappeared: the critical operation is the only one to contribute to the demand for the peak. In other words, the problem has been totally decoupled. The resource requirements of the operations that still need to be scheduled no longer interact with each other. This phenomenon is not specific to this example, but can be observed in all the problems that we have run. This suggests that the operation ordering heuristic implemented in Micro-Boss is indeed very effective at redirecting search towards the most serious conflicts.

Notice also that no backtracking was necessary to schedule this problem. The resulting schedule is displayed in Figure 8.

Figure 7 should be about here.

Conclusion of Figure 7 should be about here.

Figure 8 should be about here.

---

[13]Remember that the demand peak corresponds to the interval of highest contention in the current search state

4 Reactive and Interactive Scheduling in Micro-Boss

Manufacturing is a process often fraught with contingencies and subject to a multitude of constraints and preferences that are not always easily amenable to representation in a computer model.

Operation durations tend to vary, machines break down, raw materials fail to arrive on time, new customer orders arrive, others get cancelled, etc. Many ad-hoc constraints and preferences that vary over time such as the preference of a worker on a specific day to perform more demanding tasks in the morning may be best accounted for via interactive manipulation of the schedule. This section briefly outlines reactive and interactive scheduling capabilities currently under development in the Micro-Boss decision support system.

4.1 Reactive Scheduling and Control Issues

Small disruptions such as minor deviations in operation durations often do not warrant major modifications to the schedule. However, as the impact of small disruptions accumulate or as more severe disruptions occur, such as long machine breakdowns, it is sometimes desirable to reoptimize the schedule from a more global perspective. Accordingly, in Micro-Boss, schedule disruptions can be handled at two levels based on their severity and the required response time:

1. **Control level**: Small disruptions that require fast responses are handled by simple control heuristics such as *"process the operation with the earliest scheduled start time first"*, or, *"when a machine is down, reroute critical jobs to equivalent machines, if any"*.

2. **Scheduling level**: In the face of more severe deviations from the schedule, the control level calls upon the Micro-Boss scheduling module to repair/reoptimize the schedule from a more global perspective, while possibly continuing to attend to immediate decisions.

Determining when disruptions should be handed over to the scheduling level can be tricky. Decisions at the control level tend to be rather fast as they are based on local heuristics with a very restricted view of the problem. Decisions at the scheduling level tend to produce better repairs but take longer as they are based on more global considerations. There is generally a tradeoff between the responsiveness of the overall system and the amount of reoptimization that can be performed. In manufacturing environments where disruptions are very frequent, a large number of disruptions may need to be handled at the control level, whereas, in less chaotic environments, a larger proportion of disruptions may be processed at the scheduling level. A similar two-tier approach to handling schedule disruptions was first proposed by Smith et al. [35]. Within this approach, the scheduling level restricts the set of alternatives to be considered at the control level by imposing a legal temporal window of execution on each operation. If the controller cannot respect an operation's window of execution, it has to request a new schedule (and a new set of execution windows) from the scheduler. One objective of ongoing research in reactive scheduling and control within Micro-Boss aims at assessing the merits of different coordination regimes between the scheduling and control levels.

Schedule repair in Micro-Boss differs from recent approaches that emphasized the use of iterative repair heuristics [36, 20, 43]. In the process of resolving schedule conflicts, iterative repair heuristics are allowed to introduce new conflicts, which in turn require more repairs. This iterative behavior may sometimes lead to myopic decisions and can potentially become expensive. In contrast to these approaches, schedule repair in Micro-Boss attempts to take a more global view of the repair problem and capitalize on the strengths of the micro-opportunistic search procedures in the system. Concretely, schedule repair in Micro-Boss is performed in two steps: (1) a set of operations that need to be rescheduled is identified and all the operations in this set are unscheduled, (2) the scheduling problem consisting of all these unscheduled operations and the constraints imposed on these operations by operations that have already been executed or have not been unscheduled is passed to the micro-opportunistic scheduling module described in the previous sections. The set of operations unscheduled in the first phase is selected in such a way that the resulting scheduling problem (i.e. the one solved in phase (2)) generally admits a solution. In the event that a feasible schedule could not be built in phase (2), the system needs to return to phase (1) and undo a larger number of operations. In practice, this situation can generally be avoided by unscheduling slightly more operations than apparently required. While the resulting search space is slightly larger and hence may require longer to be explored, it may also contain better repair solutions. We are currently studying different conflict propagation heuristics to determine which (and how many) operations to reschedule to recover from different schedule disruptions[14].

4.2 Interactive Scheduling with Micro-Boss

While the combinatorial complexity of factory scheduling problems is best handled by automatic scheduling procedures such as the ones described earlier in this paper, ad-hoc scheduling constraints and preferences that occur very infrequently or change over time are often best accounted for through interactive manipulation of the schedule. Interactive scheduling techniques can also be designed to help the end-user weigh different alternatives (e.g. to decide whether or not to work overtime or work a third shift on a group of machines).

The Micro-Boss decision support system enables the end-user to interleave both manual and automatic (micro-opportunistic) scheduling decisions, edit, save, and compare complete and partial schedules.

Interactive schedule manipulation is performed using an interactive Gantt chart that displays each resource along with the operations to which that resource has been allocated over time (Figure 9). Schedule manipulation is performed under the supervision of the Micro-Boss consistency enforcing module, which enforces consistency with earlier scheduling decisions (manual and automatic). Partial or complete schedules can be saved and compared against each other along different metrics, including total schedule cost, average weighted tardiness, average weighted earliness, Work-In-Process, Work-In-System (which accounts for both Work-In-Process inventory and finished goods inventory), etc. Optimistic estimates are used for partial schedules for which these metrics cannot be computed exactly. By interleaving both manual and automatic scheduling decisions and saving/restoring partial and complete schedules, the user can compare the impact of alternative scheduling decisions and perform "what-if" analysis.

Figure 9 should be about here.

Figure 9 shows a typical view of the Micro-Boss user interface. In this example, the user is getting ready to modify the working schedule displayed in the Gantt chart, by manually unscheduling an operation on which he/she just clicked. Statistics on the working schedule are

---

[14]For particularly severe schedule disruptions such as the breakdown of a bottleneck machine over a long time period, we are also considering rescheduling techniques that subdivide the scheduling horizon and only reschedule those operations that are expected to fall within the near future while overlooking conflicts with operations whose execution is expected to take place later.

compared against statistics for the "current" schedule, namely the schedule currently in force in the system. These statistics are continuously updated as the user edits the schedule. In another window, the user can check information about specific orders (*order2* in this example). In yet another window, he/she has elected to rank orders based on their tardiness in the working schedule. Alternative metrics to rank jobs or resources can be selected in the statistics menu (e.g. cost, tardiness, flowtime, resource utilization, etc.). By clicking on boxes displayed in the Gantt chart, the user can directly obtain information on specific operations (e.g. information on operation *milling31*), manually unschedule and reschedule operations (by moving the corresponding box in the Gantt chart), unschedule jobs or highlight a job by changing its color. The Gantt menu also allows for zooming in and out of the Gantt chart, unscheduling specific resource/time intervals, displaying contention measures over time for different resources, etc.

5 Performance Evaluation

Experimental studies performed with an initial version of Micro-Boss implemented using Knowledge Craft were reported in [30]. These experiments studied the performance of the system under a variety of scheduling conditions and different cost assumptions. They included comparisons against combinations of popular priority dispatch rules and release policies advocated in the Operations Research literature, comparisons against coarser bottleneck-centered approaches to scheduling described in the Artificial Intelligence literature and a comparison against a variation of Micro-Boss in which resource contention was measured using unbiased demand profiles.

In this paper, we report the results of a similar study performed on the same set of scheduling problems with a more recent version of the system written in C++. At the present time (January 1993), the new version of Micro-Boss is about 30 times faster than the version described in [30], mainly due to the C++ reimplementation. The new system also uses a more powerful consistency enforcing procedure (See Subsection 2.2) than the original version, which almost eliminates the need for backtracking on the experiments reported in this paper. Finally, on average, the new system produces schedules that are about 10% better than those obtained with the earlier version. This improvement in schedule quality is mainly attributed to the use of a more accurate set of propagation heuristics to update the best remaining start time(s) of unscheduled operations during construction of the schedule.

Table 2 should be about here.

The results reported below were obtained on a suite of 80 scheduling problems initially described in [30]. The suite is made of eight sets of scheduling problems obtained by adjusting three parameters to cover a wide range of scheduling conditions. The three parameters are the following: an average due date parameter (tight versus loose average due date), a due date range parameter (narrow versus wide range of due dates), and a parameter controlling the number of major bottlenecks (in this case one or two). For each parameter combination, a set of 10 scheduling problems was randomly generated (see Table 2), thereby resulting in a total of 80 scheduling problems (10 problems x 2 average due date values x 2 due date ranges x 2 bottleneck configurations). Each problem requires scheduling 20 jobs on 5 resources for a total of 100 operations. Marginal tardiness costs in these problems were set to be, on the average, five times larger than marginal inventory costs to model a situation where tardiness costs dominate but inventory costs are non-negligible[15]. A comprehensive description of these scheduling problems can be found in [30].

Micro-Boss required between 20 and 30 CPU seconds to schedule each problem on a DECstation 5000. With the exception of one problem for which the system had to generate two extra search states (i.e 102 search states instead of 100), all problems were solved without backtracking.

5.1 Comparison Against Combinations of Priority Dispatch Rules and Release Policies.

In a first set of experiments, Micro-Boss was compared against the best of a set of 35 combinations of popular priority dispatch rules and release policies. The priority dispatch rules used in these experiments were of two types:

1. A set of five priority dispatch rules that have been reported to be particularly good at reducing tardiness under various scheduling conditions [38]: the Weighted Shortest Processing Time (WSPT) rule, the Earliest Due Date (EDD) rule, the Slack per Remaining Processing Time (S/RPT) rule, and two parametric rules, the Weighted Cost OVER Time (WCOVERT) rule and the Apparent Tardiness Cost (ATC) rule.

2. An exponential version of the parametric early/tardy dispatch rule recently developed by Ow and Morton [26, 21] and referred to below as EXP-ET. This rule differs from the other 5 in that it can explicitly account for both tardiness and inventory costs.

EXP-ET was run in combination with an intrinsic release policy that only releases jobs when their priorities become positive, as suggested in [21]. The other five dispatch rules were successively run in combination with two release policies: an immediate release policy (IM-REL) that allowed each job to be released immediately and the Average Queue Time release

[15]Experiments under different cost assumptions were also reported in [30].

policy (AQT) described in [21]. AQT is a parametric release policy that estimates queuing time as a multiple of the average job duration (the look-ahead parameter serving as the multiple). The release of a job is determined by offsetting the its due date by the sum of the job's total duration and the estimated queuing time. Combinations of release policies and dispatch rules with a look-ahead parameter were successively run with four different parameter values that generally appeared to produce the best schedules. By combining these different dispatch rules, release policies and parameter settings a total of 35 heuristics[16] was obtained. On each problem, the best of the 35 schedules produced by these heuristics was compared against the schedule obtained by Micro-Boss. Among the 35 scheduling heuristics (i.e. excluding Micro-Boss), each of the six dispatch rules (WSPT, EDD, S/RPT, WCOVERT, ATC and EXP-ET) and each of the three release policies (IM-REL, AQT and EXP-ET's intrinsic release policy) performed best on at least one problem out of the 80 and 12 combinations out of the 35 performed best on at least one problem.

Figure 10 should be about here.

Figure 10 compares the average cost of the schedules produced by Micro-Boss against the average cost obtained by the best of the 35 combinations of dispatch rules and release policies on each problem set. Schedule cost was computed as the sum of tardiness and inventory costs, as specified in Equation (3). The results indicate that Micro-Boss consistently outperformed the combination of 35 heuristics under all eight conditions of the study. A more detailed analysis of the results indicates that, while performing at a level comparable to the combination of dispatch rules and release policies with respect to tardiness, Micro-Boss yielded significant reductions in inventory costs. In fact, the most important reductions in inventory were observed on the most difficult problems, namely those with tight average due dates and narrow due date ranges. Overall, Micro-Boss reduced average schedule cost by more than 12% compared to the combinations of dispatch rules and release policies.

---

[16]The 35 combinations were as follows: EXP-ET and its intrinsic release policy (times four parameter settings) , EDD/AQT (times four parameter settings), EDD/IM-REL, WSPT/AQT (times four parameter settings), WSPT/IM-REL, S/RPT/AQT (times four parameter settings), S/RPT/IM-REL, WCOVERT/IM-REL (times four parameter settings), WCOVERT/AQT (times four parameter settings), ATC/IM-REL (times four parameter settings), ATC/AQT (times four parameter settings).

5.2 Comparison Against Coarser Opportunistic Scheduling Procedures

Micro-Boss was also compared against several coarser opportunistic schedulers that dynamically combined both a resource-centered perspective and a job-centered perspective, like in the OPIS scheduling system [24]. While OPIS relies on a set of repair heuristics to recover from inconsistencies [25], the macro-opportunistic schedulers of this study were built to use the same consistency enforcing techniques and the same backtracking scheme as Micro-Boss[17]. The macro-opportunistic schedulers also used the same demand profiles as Micro-Boss. When average demand for the most critical resource/time interval was above some threshold level (a parameter of the system that was empirically adjusted), the macro-opportunistic scheduler focused on scheduling the operations requiring that resource/time interval; otherwise it used a job-centered perspective to identify a critical job and schedule some or all the operations in that job. Each time a resource/time interval or a portion of a job was scheduled, new demand profiles were computed to decide which scheduling perspective to use next.

Figure 11 should be about here.

Figure 11 summarizes the results of a comparison between Micro-Boss and two macro-opportunistic schedulers that differed in the number of operations that they were allowed to schedule at once in their resource-centered perspective (referred to below as the granularity of the scheduler). The macro-opportunistic scheduler with granularity 4 was allowed to schedule up to 4 operations in its resource-centered perspective, after which it had to compute new demand profiles and decide which subproblem (job-centered or resource-centered) to focus on next. The macro-opportunistic scheduler with granularity 8 was allowed to schedule at once up to 8 operations in its resource-centered perspective. The results in Figure 11 not only indicate that Micro-Boss consistently produced better schedules than the two macro-opportunistic schedulers. They also clearly show that schedule performance degraded as the granularity of the macro-opportunistic scheduler was increased, namely as the search procedure became less flexible. More detailed performance measures not presented here indicate that the reductions in schedule cost achieved by Micro-Boss correspond to reductions in both tardiness and inventory costs.

[17] An alternative would have been to implement a variation of Micro-Boss using the same repair heuristics as OPIS. Besides being time-consuming to implement, such a comparison would have been affected by the quality of the specific repair heuristics currently implemented in the OPIS scheduler.

Overall, these results strongly suggest that the additional flexibility of a micro-opportunistic scheduling procedure over coarser opportunistic procedures generally yields important improvements in schedule quality.

5.3 Evaluating the Impact of Using Biased Demand Profiles

A third set of experiments was carried out to test the effect of using biased demand profiles to guide the micro-opportunistic scheduler. A variation of Micro-Boss using unbiased demand profiles was run on the same set of 80 scheduling problems.

Figure 12 should be about here.

Figure 12 compares the average schedule costs obtained by both variations of Micro-Boss. In seven out of the eight scheduling situations of the study, biasing the demand profiles produced reductions in schedule cost ranging from 3 to 22 percent, including an impressive 20 percent in the most difficult scheduling situation (Problem Set 8 with two bottlenecks, a tight average due date and a narrow range of due dates). In the one case (out of eight) where the unbiased version produced better schedules, the biased version was only 5% worse. A more detailed analysis of the results indicates that, overall, the biased version of Micro-Boss performed 30% better with respect to tardiness while incurring a slight increase of 0.6% in inventory costs. Altogether, biasing the demand profiles reduced schedule costs by more than 15%. These results validate both the idea of building biased demand profiles to guide the micro-opportunistic search procedure and the particular technique used in Micro-Boss to operationalize this idea (namely the use of the *mincost* functions). In general, it should be possible to obtain even better results by varying the bias according to specific problem characteristics. One could also consider fine-tuning the bias during the construction of the schedule.

6 Concluding Remarks

Current computer solutions to production management such as the one implemented in MRP/MRP-II systems are of limited help, as they rely on oversimplified models of the plant and only provide weak feedback loops to update the production schedule during execution (typically, complete updates of the schedule are only performed on a weekly basis). A major challenge for researchers in production scheduling is to come up with new techniques that can account more precisely for actual manufacturing objectives and constraints, including execution contingencies such as machine breakdowns, new job arrivals, variations in processing times, yields, etc. New production scheduling tools should also enable the user to interactively perform "what-if" analysis and account for ad-hoc constraints and/or preferences that are not easily amenable to representation in the computer model.

In this paper, we presented Micro-Boss, a decision support system for factory scheduling. Micro-Boss aims at combining powerful predictive, reactive, and interactive scheduling capabilities. To this end, the system relies on a new micro-opportunistic search procedure that enables the scheduling system to continuously track the evolution of micro-bottlenecks (or conflicts) during the construction or repair of the schedule and refocus its optimization effort on those micro-bottlenecks that appear most critical. This approach differs from earlier opportunistic approaches [24, 6], as it does not require scheduling large resource subproblems or large job subproblems before revising the current scheduling strategy. The results of an experimental study comparing Micro-Boss against combinations of popular priority dispatch rules and release policies advocated in the Operations Research literature as well as coarser opportunistic scheduling approaches proposed in the Artificial Intelligence literature suggest that the flexibility of this new search procedure can often yield important improvements in schedule quality. We find that, because of their flexibility, micro-opportunistic scheduling procedures are also particularly well suited for repairing schedules in the face of execution contingencies and can easily be integrated in interactive decision support systems that enable the user to incrementally manipulate and compare alternative schedules.

While our work on Micro-Boss has focused on generalized versions of the job shop scheduling problem, micro-opportunistic scheduling techniques have been applied to other manufacturing problems and other classes of problems such as transportation scheduling. Rautaruukki Oy, a large Finnish steel manufacturer, and researchers at the Helsinki University of Technology have reported adapting an earlier version of our micro-opportunistic scheduling heuristics to schedule a steel rolling mill [37]. Variations of the Micro-Boss scheduling heuristics are also used in the Knowledge Based Logistics Planning Shell (KBLPS) developed by Carnegie Group, Inc. (CGI) and LB&M Associates to solve U.S. army transportation scheduling problems and ammunition distribution planning problems [8, 5, 32]. Other efforts using variations of the micro-opportunistic techniques developed in the context of Micro-Boss are described in [4, 18, 28] and [42].

Current research efforts within our project aim at applying and extending the existing approach to solve both manufacturing and transportation scheduling problems.

Acknowledgement

I wish to thank Mark Fox for his help and support during the initial development of the system. Special thanks to Bob Schnelbach for his help designing and implementing the Micro-Boss user interface and reactive scheduling component, and to Shinichi Otsuka for his help with the C++ reimplementation of the system. Recent research efforts have also benefited from the participation of Bryan Lewis, Gang Li, Yoichiro Nakakuki, Katia Sycara, Joe Toomey, Sam Thangiah, and Yalin Xiong.

References

[1] J. Adams, E. Balas, and D. Zawack.
The Shifting Bottleneck Procedure for Job Shop Scheduling.
Management Science 34(3):391-401, 1988.

[2] Kenneth R. Baker and Gary D. Scudder.
Sequencing with Earliness and Tardiness Penalties: A Review.
Operations Research 38(1):22-36, January-February, 1990.

[3] Bean, J.C.,J.R. Birge, J. Mittenthal, C.E. Noon.
Matchup Scheduling with Multiple Resources, Release Dates and Disruptions.
Operations Research 39(3):470-483, May-June, 1991.

[4] Pauline M. Berry.
The PCP: A Predictive Model for Satisfying Conflicting Objectives in Scheduling Problems.
Technical Report, Centre Universitaire d'Informatique, Universite de Geneve, 12, Rue du Lac, CH-1207, Geneva, Switzerland, 1991.

[5] Camden, R., Dunmire, C., Goyal, R., Sathi,N., Elm, B., and Fox, M.
Distribution Planning: An Integration of Constraint Satisfaction and Heuristic Search Techniques.
In *Proceedings of the Conference on AI Applications in Military Logistics*. 1990.

[6] A. Collinot, C. Le Pape and G. Pinoteau.
SONIA: a Knowledge-based Scheduling System.
International Journal of Artificial Intelligence in Engineering 2(4):86-94, 1988.

[7] S. Dauzere-Peres and J.B. Lasserre.
A Modified Shifting Bottleneck Procedure for Job Shop Scheduling.
Technical Report LAAS 90106, Laboratoire d'Automatique et d'Analyse des Systemes, 7, Av. du Colonel Roche, 31077 Toulouse Cedex, France, 1990.

[8] Dunmire, C., Sathi,N., Goyal, R., Fox, M., and Kott, A.
Ammunition Inventory Planning: An Integration of Configuration and Resource Allocation Techniques.
In *Proceedings of the Conference on AI Applications in Military Logistics*. 1990.

[9] Mark S. Fox.
Constraint-Directed Search: A Case Study of Job-Shop Scheduling.
PhD thesis, Department of Computer Science, Carnegie-Mellon University, 1983.

[10] R.E. Fox.
OPT: Leapfrogging the Japanese.
Just-in-time Manufacture.
In C.A. Voss,
IFS Ltd, Springer Verlag, 1987.

[11] S. French.
Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.
Wiley, 1982.

[12] M.R. Garey and D.S. Johnson.
Computers and Intractability: A Guide to the Theory of NP-Completeness.
Freeman and Co., 1979.

[13] Eliyahu M. Goldratt.
Optimized Production Timetable: Beyond MRP: Something Better is finally Here.
October, 1980
Speech to APICS National Conference.

[14] Graves, S.C.
A Review of Production Scheduling.
Operations Research 29(4):646-675, July-August, 1981.

[15] F. Robert Jacobs.
OPT Uncovered: Many Production Planning And Scheduling Concepts Can Be Applied With Or Without The Software.
Industrial Engineering 16(10):32-41, October, 1984.

[16] Claude Le Pape and Stephen F. Smith.
Management of Temporal Constraints for Factory Scheduling.
Technical Report, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1987.
also appeared in Proc. Working Conference on Temporal Aspects in Information Systems, Sponsored by AFCET and IFIP Technical Committee TC8, North Holland Publishers, Paris, France, May 1987.

[17] Gang Li and Norman Sadeh.
Single-machine Early/Tardy Scheduling Problem with Setups: A Hybrid Heuristic Approach.
Technical Report, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1993.
Working paper. Presented at the Joint National ORSA/TIMS meeting held in San Francisco, November 1-4, 1992.

[18] Theodore A. Linden.
Preference-Directed, Cooperative Resource Allocation and Scheduling.
Technical Report, Advanced Decision Systems, 1500 Plymouth St., Mountain View, CA 94043, September, 1991.

[19] A.K. Mackworth and E.C. Freuder.
The Complexity of some Polynomial Network Consistency Algorithms for Constraint Satisfaction Problems.
Artificial Intelligence 25(1):65-74, 1985.

[20] S. Minton, M.D. Johnston, A.B. Philips, P. Laird.
Solving Large-Scale Constraint Satisfaction and Scheduling Problems Using a Heuristic Repair Method.
In *Proceedings of the Eighth National Conference on Artificial Intelligence,* pages 17-24. 1990.

[21] T.E. Morton, S.R. Lawrence, S. Rajagopolan, S Kekre.
SCHED-STAR: A Price-Based Shop Scheduling Module.
Journal of Manufacturing and Operations Management :131-181, 1988.

[22] Joseph Orlicky.
Material Requirements Planning.
McGraw Hill, New York, 1975.

[23] Peng Si Ow.
Focused Scheduling in Proportionate Flowshops.
Management Science 31(7):852-869, 1985.

[24] Peng Si Ow and Stephen F. Smith.
Viewing Scheduling as an Opportunistic Problem-Solving Process.
Annals of Operations Research 12:85-108, 1988.

[25] 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.

[26] Peng Si Ow and Thomas Morton.
The Single Machine Early/Tardy Problem.
Management Science 35(2):177-191, 1989.

[27] S.S. Panwalkar and Wafik Iskander.
A Survey of Scheduling Rules.
Operations Research 25(1):45-61, January-February, 1977.

[28] Paolucci, E., Patriarca, E., Sem, M., and Gini G.
Predit: A Temporal Predictive Framework for Scheduling Systems.
In *Proceedings of the AAAI Spring Symposium on Practical Approaches to Scheduling and Planning,* pages 150-154. 1992.

[29] N. Sadeh and M.S. Fox.
Variable and Value Ordering Heuristics for Hard Constraint Satisfaction Problems: an Application to Job Shop Scheduling.
Technical Report CMU-RI-TR-91-23, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1991.

[30] Norman Sadeh.
Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling.
PhD thesis, School of Computer Science, Carnegie Mellon University, March, 1991.

[31] Norman Sadeh, Katia Sycara, and Yalin Xiong.
A Comparative Study of Backtracking Techniques for Hard Scheduling Problems.
Technical Report CMU-RI-TR-92-06, The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA 15213, 1992.

[32] Victor Saks, Al Kepner, and Ivan Johnson.
Knowledge Based Distribution Planning.
Technical Report, Carnegie Group, Inc., 5 PPG Place, Pittsburgh, PA 15222, 1992.

[33] P. Serafini, W. Ukovich, H. Kirchner, F. Giardina, and F. Tiozzo.
Job-shop scheduling: a case study.
Operations Research Models in FMS.
In F. Archetti, M. Lucertini, and P. Serafini,
Springer, Vienna, 1988.

[34] S. Smith, M. Fox, and P.S. Ow.
Constructing and Maintaining Detailed Production Plans: Investigations into the Development of Knowledge-Based Factory Scheduling Systems.
AI Magazine 7(4):45-61, Fall, 1986.

[35] Smith, S.F., N. Keng, and K. Kempf.
Exploiting Local Flexibility During Execution of Pre-Computed Schedules.
Technical Report CMU-TR-RI-90-13, The Robotics Institute, Carnegie Mellon Univeristy, June, 1990.

[36] Stephen F. Smith, Peng Si Ow, Nicola Muscettola, Jean-Yves Potvin, Dirk Matthys.
An Integrated Framework for Generating and Revising Factory Schedules.
Journal of the Operational Research Society 41(6):539-552, 1990.

[37] Seppo Torma, Ora Lassila and Markku Syrjanen.
Adapting the Activity-Based Scheduling Method to Steel Rolling.
In G. Doumeingts, J. Browne, and M Tomljanovich (editor), *Proceedings of the Fourth IFIP Conference on Computer Applications in Production and Engineering (CAPE'91)*, pages 159-166. Elsevier Science Publishers B.V. (North Holland), 1991.

[38] Ari P.J. Vepsalainen and Thomas E. Morton.
Priority Rules for Job Shops with Weighted Tardiness Costs.
Management Science 33(8):1035-1047, 1987.

[39] Thomas Vollmann, William Berry, and Clay Whybark.
Manufacturing Planning and Control.
Dow Jones-Irwin, Homewood, IL, 1988.
Second Edition.

[40] Oliver Wight.
MRP II: Unlocking America's Productivity Potential.
Oliver Wight Limited Publications, Williston, VT, 1981.

[41] Oliver Wight.
Manufacturing Resource Planning: MRPII.
Oliver Wight Limited Publications, Essex Junction, VT, 1984.

[42] Andreas Winklhofer, Manfred Maierhofer, and Paul Levi.
Efficient Propagation and Computation of Problem Features for Activity-Based Scheduling.
In *Proceedings of the Seventh Symposium on Information Control Problems in Manufacturing Technology (INCOM-92).* Toronto, Canada, 1992.

[43] Monte Zweben, Eugene Davis, and Michael Deale.
Iterative Repair for Scheduling and Rescheduling.
Technical Report, NASA Ames Reserch Center, MS 244-17, Moffett Field, CA 94035, 1991.

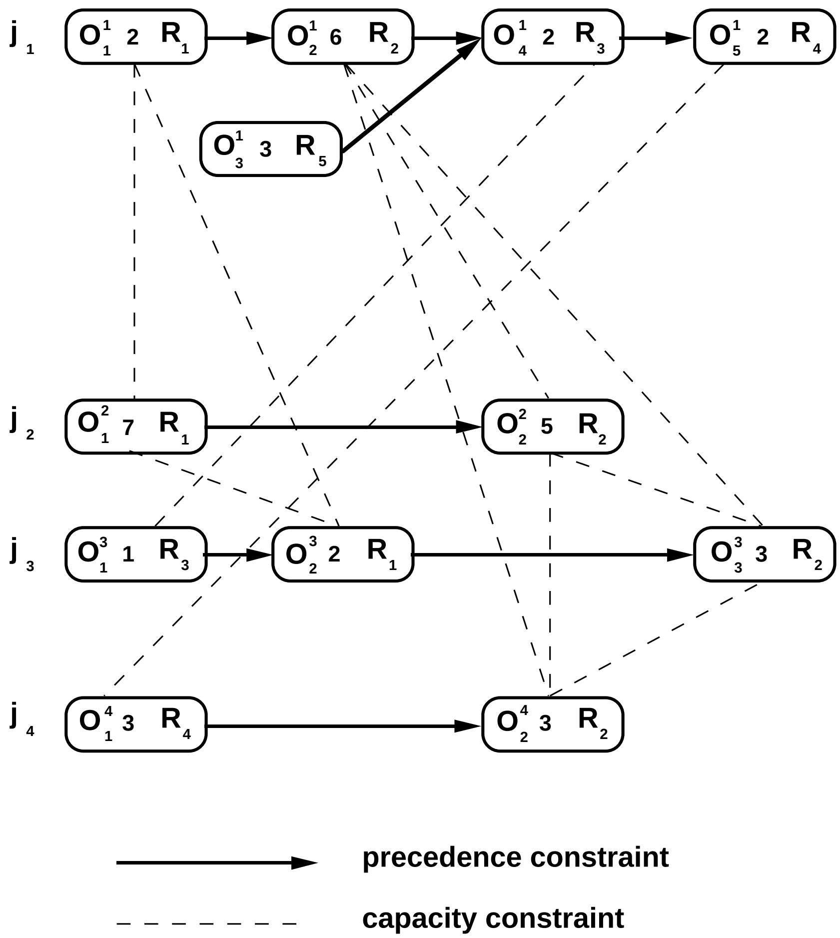

Figure 1: A simple job shop problem with 4 jobs. Each node is labeled by the operation that it represents, its duration, and the resource that it requires.

Earliest acceptable release dates, due dates, latest acceptable completion dates, and costs									
Job j_l	erd_l	dd_l	lcd_l	$tard_l$	inv_1^l	inv_2^l	inv_3^l	inv_4^l	inv_5^l
j_1	0	12	20	20	2	1	2	0	0
j_2	0	14	20	20	5	0	-	-	-
j_3	0	9	20	5	1	0	0	-	-
j_4	0	18	20	10	1	0	-	-	-

Table 1: Earliest acceptable release dates, due dates, latest acceptable completion dates and marginal costs.

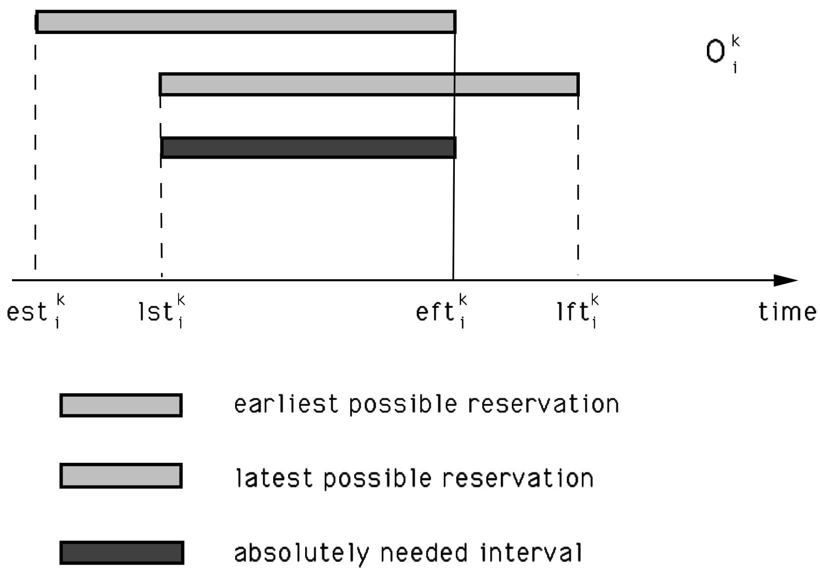

Figure 2: An example of an unscheduled operation that absolutely needs a resource/time interval.

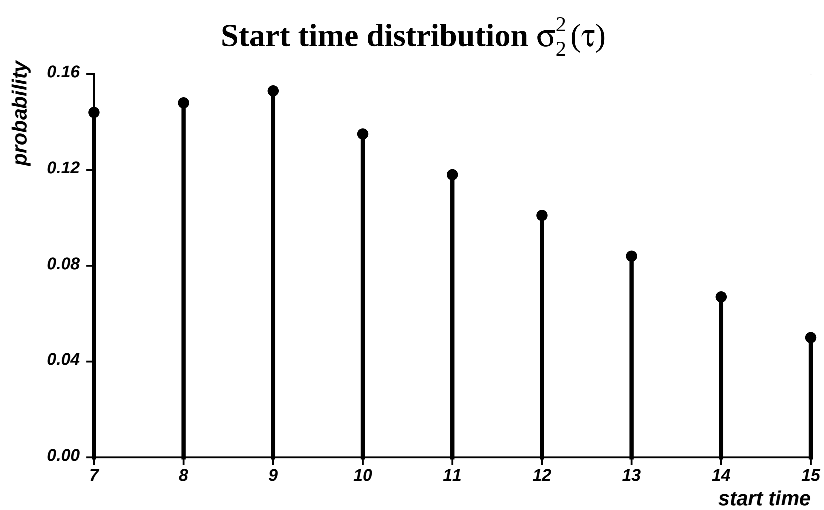

Figure 3: Start time distribution $\sigma_2^2(\tau)$ for operation O_2^2 in the initial search state for the problem defined in Figure 1.

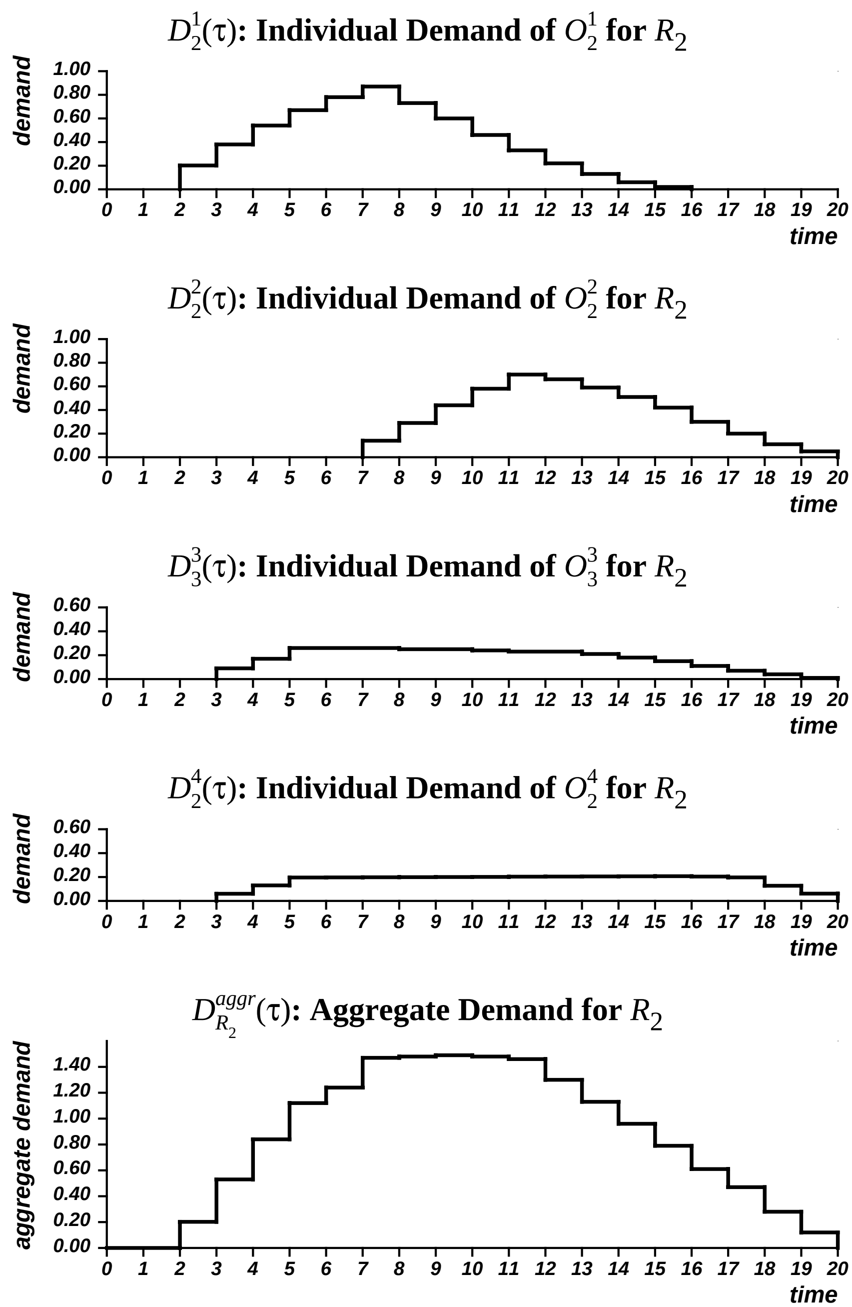

Figure 4: Building R_2's aggregate demand profile in the initial search state.

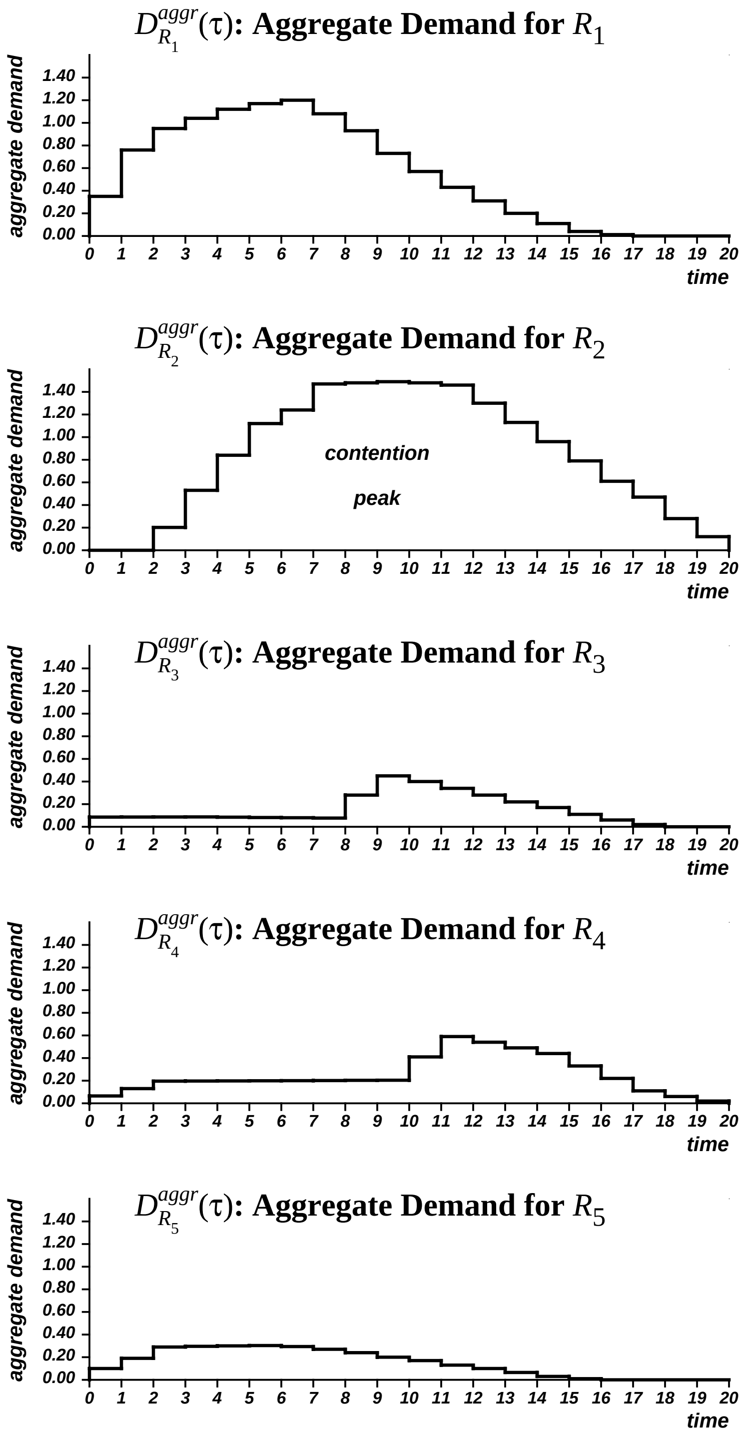

Figure 5: Aggregate demands in the initial search state for each of the five resources.

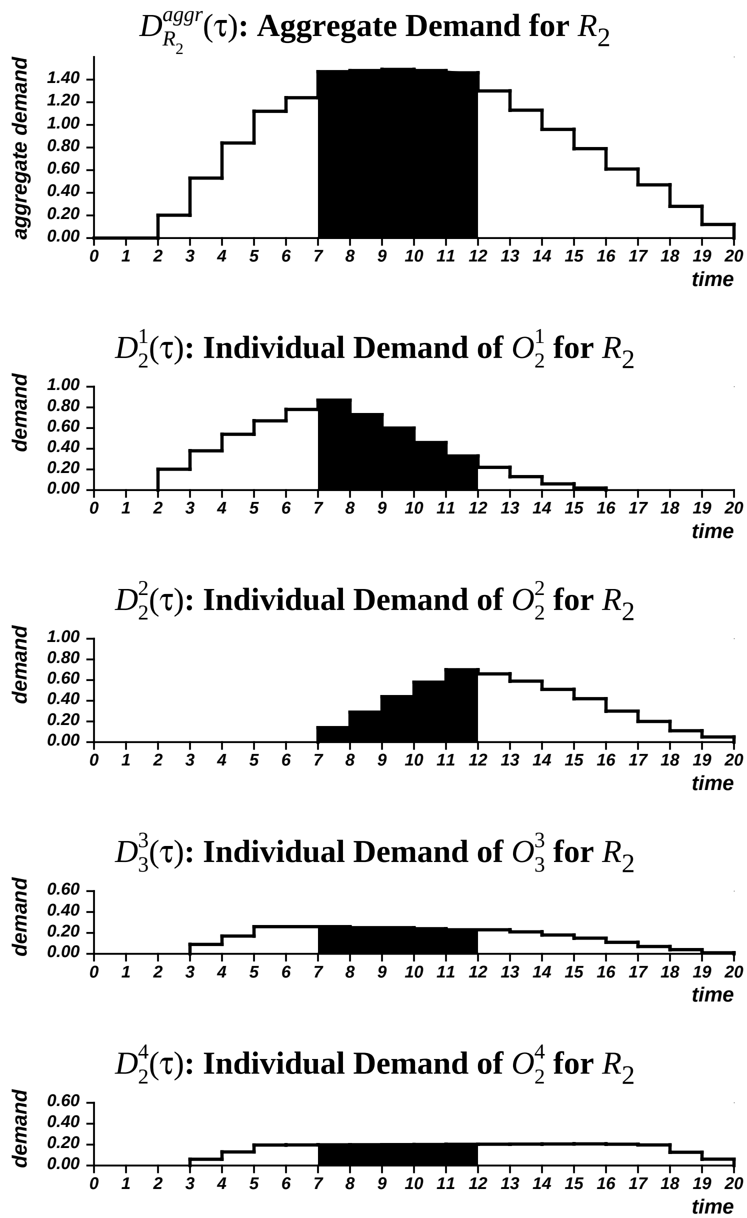

Figure 6: Operation selection in the initial search state.

MON NOV 12 1990 --- 17:04:49 EST

>> Depth: 0, Number of states visited: 0
Critical demand peak:
R_2 **between 7 and 12, Avg. expected demand: 1.48**
Critical Operation: O_2^1**, Avg. contrib.: 0.60**
Using early/tardy reservation ordering heuristic:
O_2^1 **scheduled between 2 and 8 on** R_2

>> Depth: 1, Number of states visited: 1
Critical demand peak:
R_2 **between 10 and 15, Avg. expected demand: 1.33**
Critical Operation: O_2^2**, Avg. contrib.: 0.64**
Using early/tardy reservation ordering heuristic:
O_2^2 **scheduled between 9 and 14 on** R_2

>> Depth: 2, Number of states visited: 2
Critical demand peak:
R_1 **between 0 and 4, Avg. expected demand: 1.35**
Critical Operation: O_1^2**, Avg. contrib.: 0.75**
Using early/tardy reservation ordering heuristic:
O_1^2 **scheduled between 2 and 9 on** R_1

>> Depth: 3, Number of states visited: 3
Critical demand peak:
R_2 **between 14 and 19, Avg. expected demand: 1.13**
Critical Operation: O_3^3**, Avg. contrib.: 0.58**
Using early/tardy reservation ordering heuristic:
O_3^3 **scheduled between 17 and 20 on** R_2

>> Depth: 4, Number of states visited: 4
Critical demand peak:
R_2 **between 14 and 19, Avg. expected demand: 0.60**
Critical Operation: O_2^4**, Avg. contrib.: 0.60**
Using greedy reservation ordering heuristic:
O_2^4 **scheduled between 14 and 17 on** R_2

>> Depth: 5, Number of states visited: 5
Critical demand peak:
R_4 **between 10 and 13, Avg. expected demand: 0.57**
Critical Operation: O_5^1**, Avg. contrib.: 0.34**
Using greedy reservation ordering heuristic:
O_5^1 **scheduled between 10 and 12 on** R_4

Figure 7: An edited trace

>> Depth: 6, Number of states visited: 6
Critical demand peak:
R_3 **between 8 and 10, Avg. expected demand: 1.08**
Critical Operation: O_4^1**, Avg. contrib.: 1.0**
Using greedy reservation ordering heuristic:
O_4^1 **scheduled between 8 and 10 on** R_3

>> Depth: 7, Number of states visited: 7
Critical demand peak:
R_5 **between 4 and 7, Avg. expected demand: 0.55**
Critical Operation: O_3^1**, Avg. contrib.: 0.55**
Using greedy reservation ordering heuristic:
O_3^1 **scheduled between 5 and 8 on** R_5

>> Depth: 8, Number of states visited: 8
Critical demand peak:
R_1 **between 0 and 4, Avg. expected demand: 0.50**
Critical Operation: O_1^1**, Avg. contrib.: 0.50**
Using greedy reservation ordering heuristic:
O_1^1 **scheduled between 0 and 2 on** R_1

>> Depth: 9, Number of states visited: 9
Critical demand peak:
R_4 **between 5 and 8, Avg. expected demand: 0.44**
Critical Operation: O_1^4**, Avg. contrib.: 0.44**
Using greedy reservation ordering heuristic:
O_1^4 **scheduled between 7 and 10 on** R_4

>> Depth: 10, Number of states visited: 10
Critical demand peak:
R_1 **between 12 and 16, Avg. expected demand: 0.31**
Critical Operation: O_2^3**, Avg. contrib.: 0.31**
Using greedy reservation ordering heuristic:
O_2^3 **scheduled between 15 and 17 on** R_1

>> Depth: 11, Number of states visited: 11
Critical demand peak:
R_3 **between 13 and 15, Avg. expected demand: 0.14**
Critical Operation: O_1^3**, Avg. contrib.: 0.14**
Using greedy reservation ordering heuristic:
O_1^3 **scheduled between 14 and 15 on** R_3

>> Depth: 12, Number of states visited: 12
Schedule Completed

Figure 7, concluded

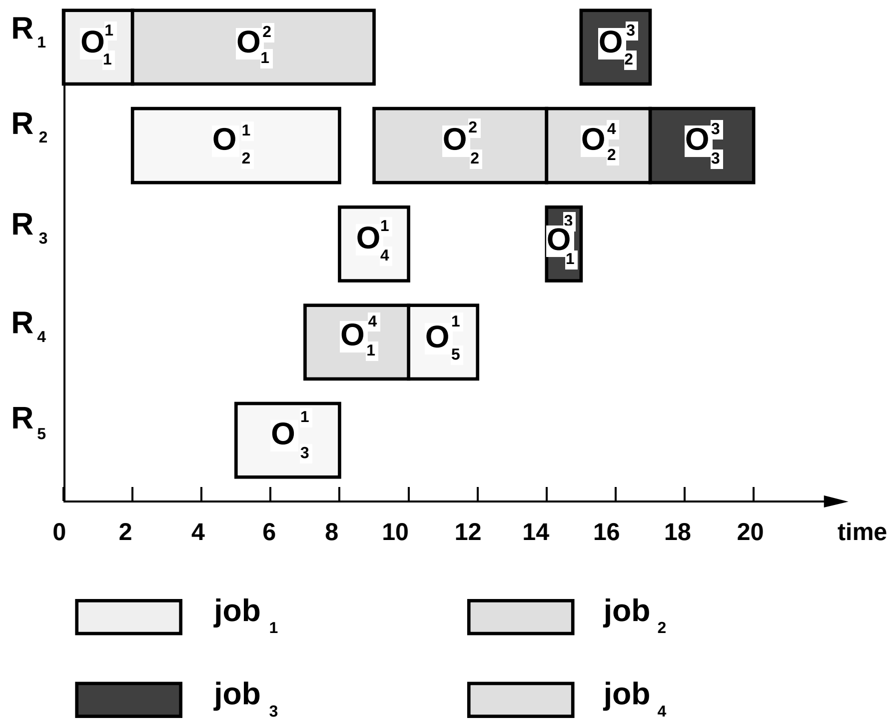

Figure 8: Gantt chart of the final schedule produced by Micro-Boss.

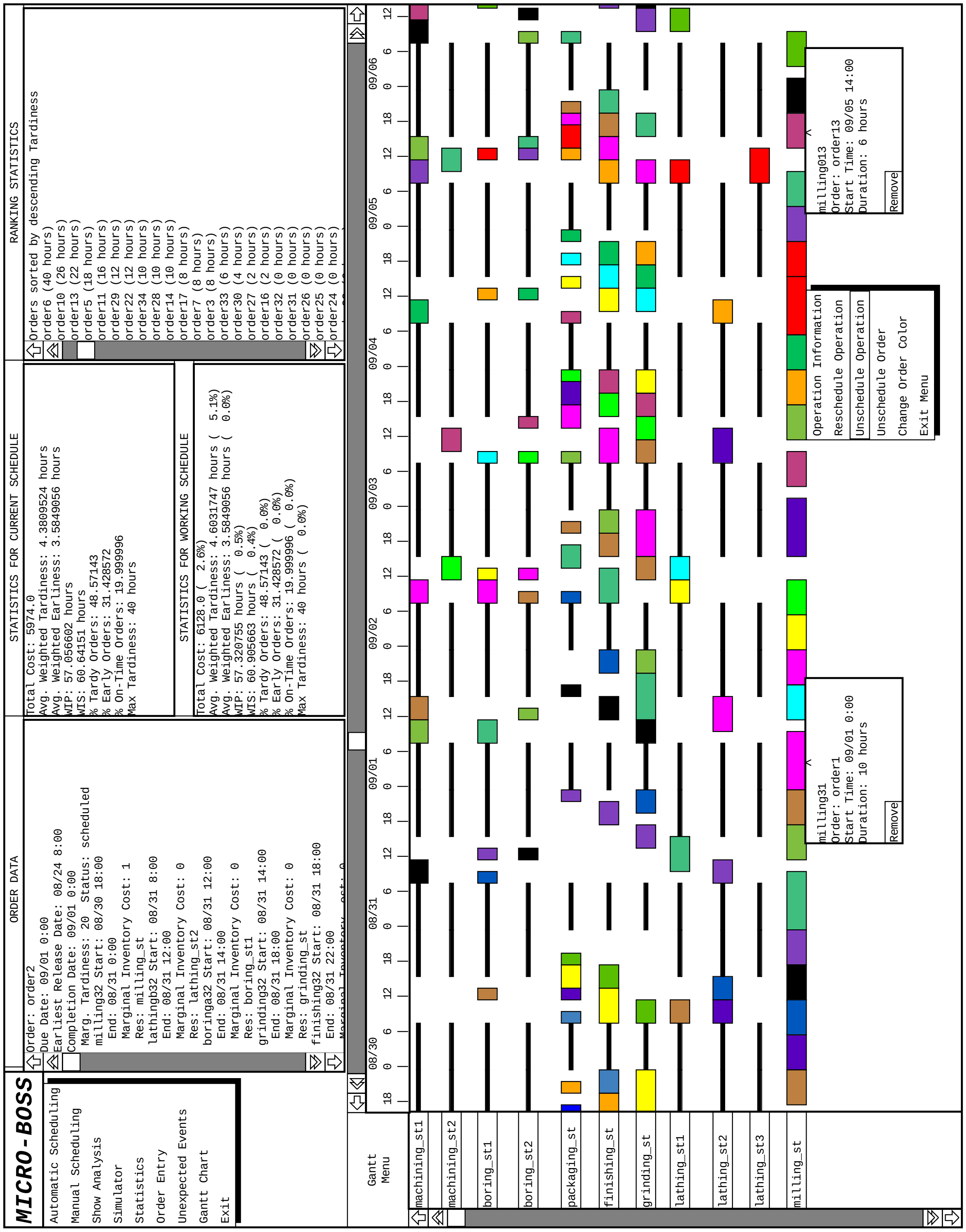

Figure 9: The Micro-Boss user interface allows for interactive manipulation of schedules. By interleaving both manual and automatic scheduling decisions, saving and comparing alternative schedules, the user can easily assess different tradeoffs and locally impose ad-hoc constraints or preferences that are not easily amenable to representation in the computer model.

Problem Sets			
Problem Set	Number of Bottlenecks	Avg. Due Date	Due Date Range
1	1	loose	wide
2	1	loose	narrow
3	1	tight	wide
4	1	tight	narrow
5	2	loose	wide
6	2	loose	narrow
7	2	tight	wide
8	2	tight	narrow

Table 2: Characteristics of the eight problem sets.

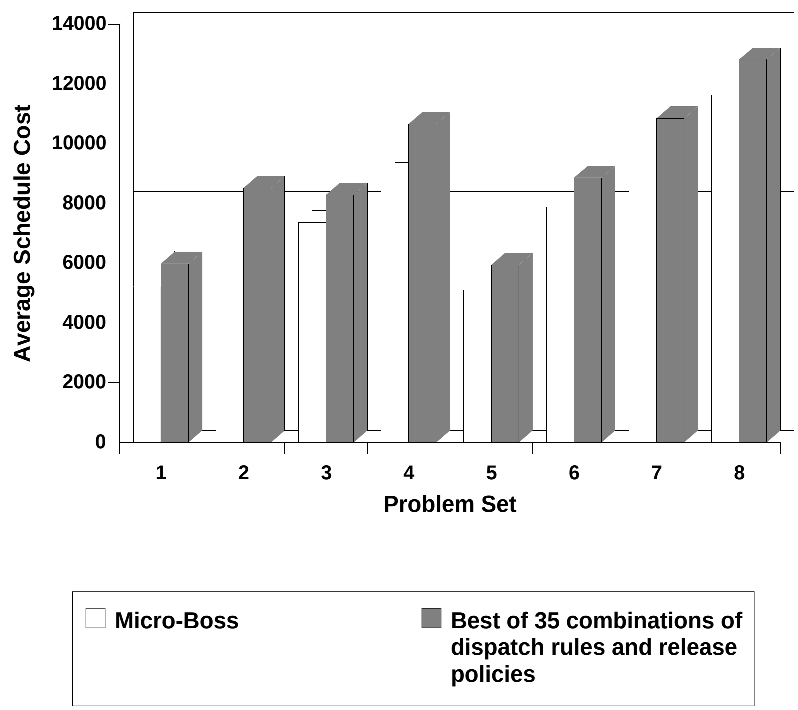

Figure 10: Comparison of Micro-Boss and the best of 35 combinations of priority dispatch rules and release policies under 8 different scheduling conditions (10 problems were generated under each condition).

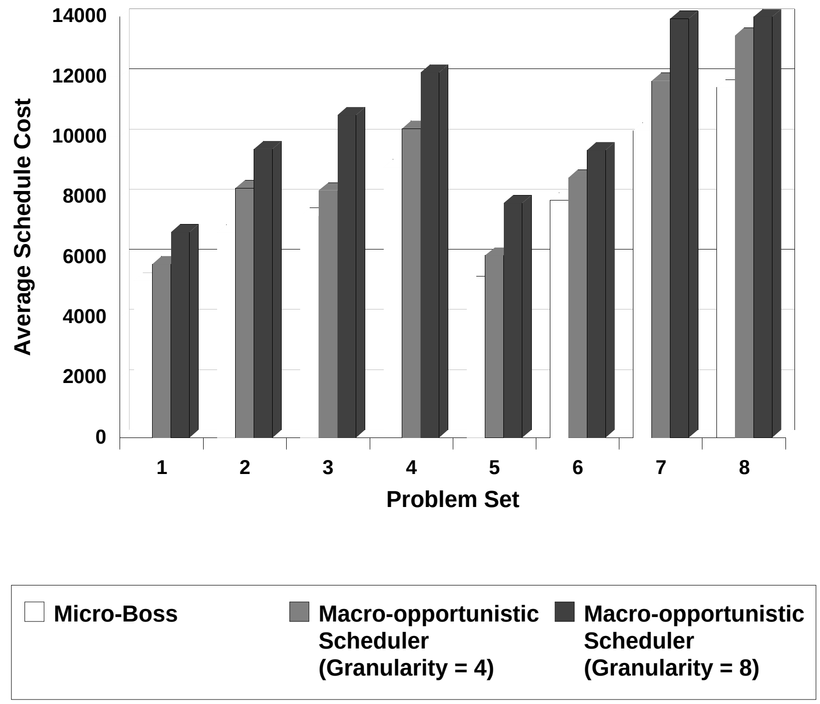

Figure 11: Comparison of Micro-Boss and two coarser opportunistic schedulers.

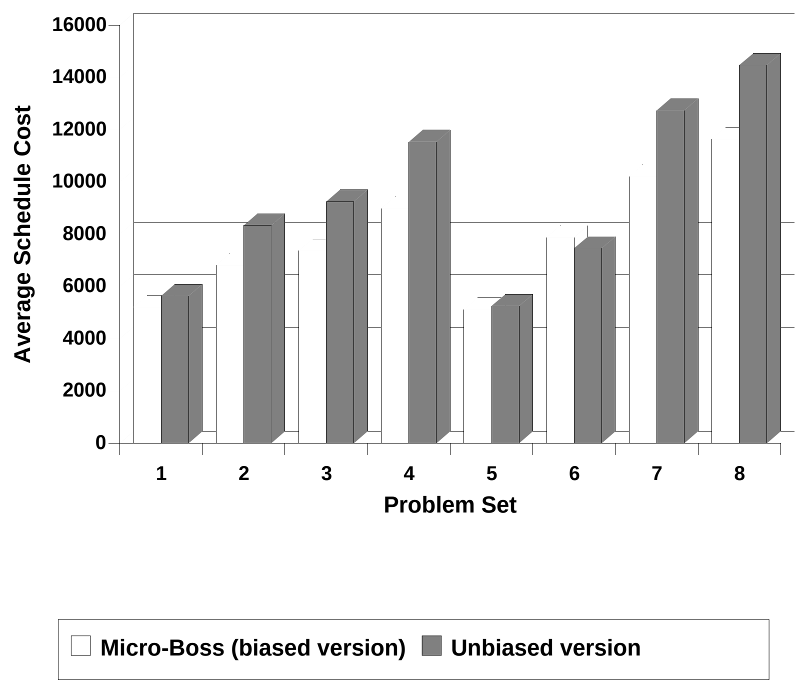

Figure 12: Comparison of the cost of the schedules produced by Micro-Boss and a variation of the system that used unbiased demand profiles.

Table of Contents

List of Figures

List of Tables