N93-32133

The MICRO-BOSS Scheduling System: Current Status and Future Efforts

Norman M. Sadeh

Center for Integrated Manufacturing Decision Systems
The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213 - U.S.A.

sadeh@ri.cmu.edu

1. INTRODUCTION

[1]Over the past few years, several approaches to scheduling have been proposed that attempt to reduce tardiness and inventory costs by *opportunistically* (i.e. dynamically) combining a resource-centered perspective to schedule bottleneck resources, and a job-centered perspective to schedule non-bottleneck operations on a job by job basis. Rather than relying on their initial bottleneck analysis, these schedulers reexamine the problem each time a resource or a job has been scheduled. This enables them to detect the emergence of new bottlenecks during the construction of the schedule. This ability has been termed *opportunistic scheduling* [3]. Nevertheless, the opportunism in these systems remained limited, as they required scheduling large resource-subproblems or large job-subproblems before allowing for a change in the scheduling perspective (i.e. before permitting a revision in the current scheduling strategy). For this reason, we actually refer to these approaches as *macro-opportunistic* techniques.

In reality, bottlenecks do not necessarily span over the entire scheduling horizon. Moreover they tend to shift before being entirely scheduled. A scheduler that can only schedule large resource subproblems will not be able to take advantage of these considerations. Often it will overconstrain its set of alternatives before having worked on the subproblems that will most critically determine the quality of the entire schedule. This in turn will often result in poorer solutions. A more flexible approach would allow to quit scheduling a resource as soon as another resource is identified as being more constraining[2]. In fact, in the presence of multiple bottlenecks, one can imagine a technique that constantly shifts attention from one bottleneck to another rather than focusing on the optimization of a single bottleneck at the expense of others. Therefore, it seems desirable to investigate a more flexible approach to scheduling, or a *micro-opportunistic* approach, in which the evolution of bottlenecks is continuously monitored during the construction of the schedule, and the problem solving effort constantly redirected towards the most serious bottleneck. 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.

Section 2 describes a micro-opportunistic factory scheduler called **MICRO-BOSS** (Micro-Bottleneck Scheduling System). Section 3 describes an empirical study that compares

[1]This research was supported, in part, by the Defense Advanced Research Projects Agency under contract #F30602-88-C-0001, and in part by grants from McDonnell Aircraft Company and Digital Equipment Corporation.

[2] [1] describes an alternative approach in which resources can be resequenced to adjust for resource schedules built further down the road. This approach has been very successful at minimizing makespan. Attempts to generalize the procedure to account for due dates seem to have been less successful so far [6].

MICRO-BOSS against a macro-opportunistic scheduler that dynamically combines both a resource-centered perspective and a job-centered perspective. A summary is provided in Section 4, along with a brief discussion of current research efforts.

2. A MICRO-OPPORTUNISTIC APPROACH

In the micro-opportunistic approach implemented in MICRO-BOSS, each operation is considered an independent decision point. Any operation can be scheduled at any time, if deemed appropriate by the scheduler. 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. start time) 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 so-called consistency enforcing procedure is applied to that state, that updates 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: the system needs to *backtrack* (i.e. it needs to undo some earlier reservation assignments in order to be able to complete the schedule). If the search state does not appear to be a deadend, the scheduler moves on and looks for a new operation to schedule and a reservation to assign to that operation.

In MICRO-BOSS, search efficiency is 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 (so-called *operation ordering heuristic*) and which reservation to assign to that operation (so-called reservation ordering heuristic).

1. **Consistency Enforcing (or Consistency Checking):** Consistency enforcing techniques prune the search space by inferring new constraints resulting from earlier reservation assignments [2, 5].
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 effort towards the most serious conflicts, the scheduler is able to build schedules that are closer to the global optimum. Simultaneously, because the scheduling strategy is aimed at reducing job contention as fast as possible, chances of backtracking tend to subside pretty fast too.

The so-called opportunism in MICRO-BOSS results from its ability 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 that displayed by other approaches where the scheduling entity is an entire resource or an entire job [3], i.e. where an entire resource or an entire job needs to be scheduled before the scheduler is allowed to revise its current scheduling strategy.

3. PERFORMANCE EVALUATION

MICRO-BOSS was compared against a variety of scheduling techniques, including popular combinations of priority dispatch rules and release policies suggested in the Operations Management literature [5].

This section outlines a study comparing MICRO-BOSS against a macro-opportunistic scheduler that dynamically combined both a resource-centered perspective and a job-centered perspective, like in the OPIS scheduling system [3]. However, while OPIS relies on a set of repair heuristics to recover from inconsistencies [4], the macro-opportunistic scheduler of this study was built to use the same consistency enforcing techniques and the same backtracking scheme as MICRO-BOSS[3]. The macro-opportunistic scheduler 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. Additional details on the implementation of the macro-opportunistic scheduler can be found in [5].

In order to compare the two schedulers, a set of 80 scheduling problems was randomly generated to cover a wide variety of scheduling conditions: tight/loose average due dates, narrow/wide due date ranges, one or two bottleneck machines. Each problem involved 20 jobs and 5 resources for a total of 100 operations (see [5] for further details).

[3]An alternative would have been to implement a variation of MICRO-BOSS using the same repair heuristics as OPIS. Besides being quite 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.

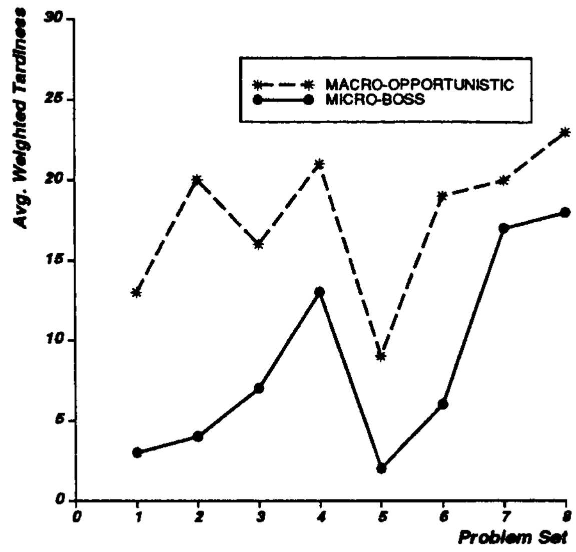

Figure 3-1: Tardiness performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

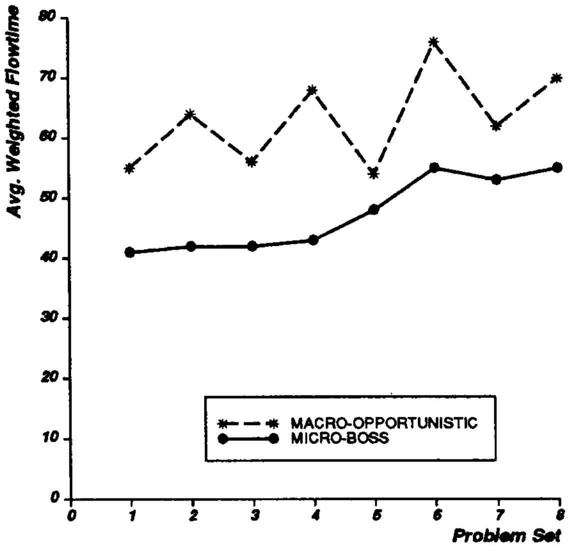

Figure 3-2: Flowtime performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

Figures 3-1, 3-2 and 3-3 summarize the results of the comparison between MICRO-BOSS and the macro-opportunistic scheduler[4]. The macro-opportunistic scheduler was consistently outperformed by MICRO-BOSS (under all eight scheduling conditions) both with

[4]The results presented in this section correspond to the 69 experiments (out of 80) that were each solved in less than 1,000 search states by the macro-opportunistic scheduler.

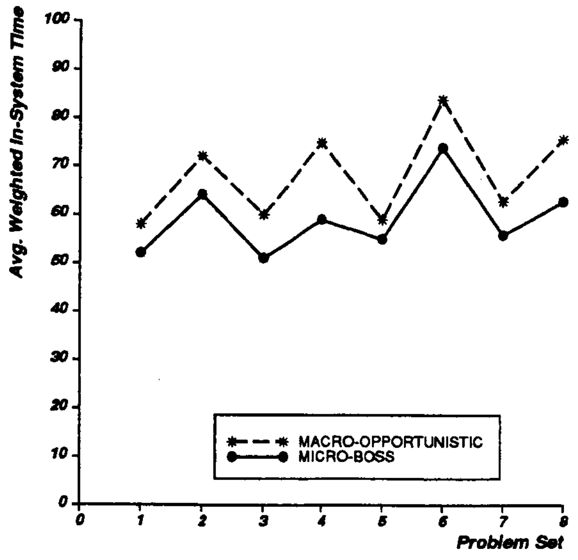

Figure 3-3: In-system time performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

respect to tardiness, flowtime (i.e. work-in-process) and in-system time (i.e. total inventory, including finished-goods inventory). *More generally, these results indicate that highly contended resource/time intervals can be very dynamic, and that it is critical to constantly follow their evolution in order to produce quality schedules.*

In most problems, MICRO-BOSS achieved a search efficiency of 100% (computed as the ratio of the number of operations to be scheduled over the number of search states that were visited), and required about 10 minutes of CPU time to schedule each problem. The current system is written in Knowledge Craft, a frame-based representation language built on top of Common Lisp, and runs on a DECstation 5000.

4. CONCLUSIONS

In this paper, a micro-opportunistic approach to factory scheduling was described that closely monitors the evolution of bottlenecks during the construction of the schedule, and continuously redirects search towards the bottleneck that appears to be most critical. This approach differs from earlier opportunistic approaches, such as the one described in [3], as it does not require scheduling large resource subproblems or large job subproblems before revising the current scheduling strategy. This micro-opportunistic approach has been implemented in the context of the MICRO-BOSS factory scheduling system. A study comparing MICRO-BOSS against a macro-opportunistic scheduler suggests that the additional flexibility of the micro-opportunistic approach to scheduling generally yields important reductions in both tardiness and inventory.

Current research efforts include:

- Adaptation of MICRO-BOSS to deal with sequence-dependent setups
- Development of micro-opportunistic reactive scheduling techniques that will enable the system to patch the schedule in the presence of contingencies such as machine breakdowns, raw materials arriving late, job cancelations, etc.

APPENDIX: PROBLEM SETS

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

REFERENCES

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

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

3. Peng Si Ow and Stephen F. Smith. "Viewing Scheduling as an Opportunistic Problem-Solving Process". *Annals of Operations Research 12* (1988), 85-108.

4. P.S. Ow, S.F. Smith, and A. Thiriez. Reactive Plan Revision. Proceedings of the Seventh National Conference on Artificial Intelligence, 1988, pp. 77-82.

5. Norman Sadeh. *Look-ahead Techniques for Micro-opportunistic Job Shop Scheduling*. Ph.D. Th., School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213, March 1991.

6. P. Serafini, W. Ukovich, H. Kirchner, F. Giardina, and F. Tiozzo. Job-shop scheduling: a case study. In *Operations Research Models in FMS*, Springer, Vienna, 1988.