Computer Applications in Production and Engineering: Integration Aspects
G. Doumeingts, J. Browne and M. Tomljanovich (Editors)
Elsevier Science Publishers B.V. (North-Holland)
© IFIP, 1991

Micro- vs. Macro-opportunistic Scheduling

Norman M. Sadeh and Mark S. Fox

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, msf@ri.cmu.edu

Abstract

[1]Over the past ten years, several approaches to scheduling have been proposed that attempt to reduce tardiness and inventory costs by *opportunistically* (i.e. dynamically) combining resource-centered heuristics (to increase resource utilization) with job-centered heuristics (to reduce inventory). Typically, these systems rely on rather coarse problem decompositions in which large resource-subproblems or large job-subproblems are scheduled at once. In contrast, this paper describes a so-called micro-opportunistic approach to scheduling which, rather than scheduling large subproblems, focuses on the scheduling of critical operations. Experimental results suggest that the flexibility of the micro-opportunistic approach to scheduling often translates into significant reductions in both tardiness and inventory over coarser scheduling approaches (referred to as macro-opportunistic approaches).

1. INTRODUCTION

The job shop scheduling problem requires scheduling a set of jobs on a finite set of resources (e.g. machines, human operators, etc.). Each job is a request for the scheduling of a set of operations according to a process routing that specifies a partial ordering among these operations, along with their resource requirements. Operations are atomic: once started they cannot be interrupted. This paper is concerned with the design of a factory scheduler that builds a detailed schedule for the jobs to be produced over a planning horizon that ranges from a couple of days to a couple of weeks. The jobs to be scheduled are provided by a master-scheduler along with due dates, earliest acceptable release dates, marginal tardiness costs, marginal in-process inventory costs (e.g. interests on raw material costs, marginal direct holding costs, interests on processing costs, etc.), and marginal finished-goods inventory costs.

[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.

Traditionally job shop scheduling problems have been solved using priority dispatcl rules [3]. These are local decision rules of the greedy type that build schedules via ɛ forward simulation of the shop. Because these rules lack a global view of the shop, the usually build up inventory in front of bottleneck machines. More recently, with th advent of more powerful computers, a couple of more sophisticated scheduling method have been developed [2, 1, 6, 5]. The OPT scheduling technique, by far the most publicize of these techniques, emphasized among other things the need to distinguish betweer bottleneck and non-bottleneck machines [2]. In OPT, an initial analysis is performed tha helps identify bottleneck resources. OPT schedules these resources first and attempts t maximize their utilization as much as possible. Non-bottleneck operations are ther scheduled so as to minimize inventory. The distinction between bottleneck and non bottleneck machines was pushed one step further in the OPIS system [6], where it wa recognized that new bottlenecks can appear during the construction of the schedule. Th OPIS scheduler combines two scheduling perspectives: a resource-centered perspective fo scheduling bottleneck resources, and a job-centered perspective to schedule non-bottlenecl 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. Thi ability to detect the emergence of new bottlenecks during the construction of the schedul has been termed *opportunistic scheduling* [6]. Nevertheless, the opportunism in thi approach remains limited as it requires scheduling large resource-subproblems or larg job-subproblems before allowing a change in the scheduling perspective (i.e. befor permitting a revision in the current scheduling strategy). For this reason, we refer t 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 onl schedule entire resources will not be able to take advantage of these considerations. Ofter it will overconstrain its set of alternatives before having worked on the subproblems tha will most critically determine the quality of the entire schedule. This in turn will ofter result in poorer solutions. A more flexible approach would allow to quit scheduling resource as soon as another resource is identified as being more constraining[2]. In fact, i the presence of multiple bottlenecks, one can imagine a technique that constantly shift attention from one bottleneck to another rather than focusing on the optimization of single bottleneck at the expense of others. Therefore, it seems desirable to investigate more flexible approach to scheduling, or a *micro-opportunistic* approach, in which th 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-centere* view of scheduling, in which each operation is considered an independent decision poin and can be scheduled without requiring that other operations using the same resource o belonging to the same job be scheduled at the same time.

Section 2 describes a micro-opportunistic factory scheduler called **MICRO-BOS** (Micro-Bottleneck Scheduling System), concentrating on the heuristics to decide whicl operation to schedule next and which reservation to assign to that operation. Section

[2] [1] describes an alternative approach in which resources can be resequenced to adjust for resourc schedules built further down the road. This approach has been very successful at minimizing makespar Attempts to generalize the procedure to account for due dates seem to have been less successful so far [10].

describes an empirical study that compares 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.

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 [4, 9].

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 [6], 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.

The remainder of this section gives a more detailed description of the look-ahead analysis step and the operation and reservation ordering heuristics. Further details on these techniques as well as other aspects of MICRO-BOSS can be found in [9].

2.1. Look-ahead Analysis in MICRO-BOSS

A two-step look-ahead procedure is applied to each search state, which 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 (within their jobs) conflict with the good reservations of other operations. Scheduling these critical operations requires finding a good compromise between the costs incurred by these operations (i.e. by the jobs to which these operations belong) and those of the operations with which they compete. Because these tradeoffs critically affect the quality of the entire schedule, they should be worked out as early as possible. This way, the scheduler ensures that it still has many options left to select judicious tradeoffs. The remaining operations also tend to become more decoupled and hence easier to optimize. By constantly redirecting search towards the most serious tradeoff operations, the scheduler produces better schedules and simultaneously reduces its chances of backtracking.

This subsection briefly describes the two-step look-ahead analysis implemented in MICRO-BOSS. Following subsections describe the operation and reservation ordering heuristics based on this look-ahead analysis.

2.1.1. Step 1: Reservation Optimization Within a Job

In addition to keeping track of the start times that remain available to an unscheduled operation, say operation O_i^k in job j_k, the scheduler implicitly maintains a cost, $mincost_i^k(\tau)$, for each of the possible start times τ of that operation. This cost, $mincost_i^k(\tau)$, is the minimum cost that would be incurred by job j_k, if O_i^k was to start at $st_i^k = \tau$ rather than at one of the best start times currently available to that operation. As other operations both within the job and in other jobs are scheduled, these costs change. Some start times become unavailable. The relative merits of start times that remain possible may change due to scheduling decisions made within the same job or in other jobs. As the scheduler moves from one search state to the other, it is critical to reassess the merits of the remaining possible start times of each unscheduled operation in order to identify where important tradeoffs still need to be made. This requires identifying whether the choice of a start time for a given unscheduled operation still affects the tardiness costs of the job to which that operation belongs. It also requires determining which inventory costs, if any, are still affected by that choice. Once these costs have been determined, the scheduler can identify which start time(s) among the ones that are still possible will be optimal for the operation (within the job), and the minimum costs that would be incurred by the job if, instead, a suboptimal start time was selected. Efficient procedures to update these costs, when going from one search state to another, are described in [9]

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

If all operations could be simultaneously scheduled at one of their best start times, there would be no scheduling problem. Generally this is not the case. Jobs typically have conflicting requirements. The micro-opportunistic approach attempts to identify critical operations whose good start times (within their jobs) conflict with the good start times of other operations, and focuses on optimizing these conflicts first. The importance of a conflict (and the criticality of the operations participating in that 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 (i.e. start times) still available to the conflicting operations and the differences in cost between these alternative reservations (as determined by the *mincost* functions computed in the previous step).

In order 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. Possible start times with lower *mincost* values are simply assigned a larger probability, thereby reflecting our expectation that they will allow for the production of better schedules. Using these start time distributions, the scheduler builds, for each unscheduled operation O_i^l, an *individual demand profile* $D_i^l(t)$ that indicates the subjective probability that the operation will be requiring its resource as a function of time (i.e. also a measure of the reliance of that operation on the availability of that resource as a function of time). By aggregating the individual demand profiles of all unscheduled operations requiring a given resource, R_k, the scheduler obtains an *aggregate demand profile*, $D_{R_k}^{aggr}(t)$, that indicates contention between (all) unscheduled operations for that resource as a function of time.

2.2. Operation Selection

Critical operations are identified as operations whose good reservations 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, and the operation with the largest reliance on the availability of that peak (i.e. the operation with the largest individual contribution to the peak) is selected to be scheduled next. Intuitively the operation that relies most on the availability of the most contended resource/time interval is also the one whose good reservations are the most likely to become unavailable if other operations contending for that resource/time interval were scheduled first.

2.3. Reservation Selection

Once it has selected an operation, MICRO-BOSS attempts to identify a reservation for that operation that will minimize as much as possible the costs incurred by the job to which that operation belongs and by other competing jobs. This is equivalent to solving a one-machine early/tardy 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).

MICRO-BOSS uses a hybrid reservation ordering heuristic[3] that adapts to the amount of contention for the critical resource/time interval. When contention is particularly high, a variation of the early/tardy procedure described in [8] is used to identify a good reservation for the operation to be scheduled. When contention is lower, the scheduler 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).

3. PERFORMANCE EVALUATION

MICRO-BOSS was compared against a macro-opportunistic scheduler that dynamically combined both a resource-centered perspective and a job-centered perspective, like in the OPIS scheduling system [6]. However, while OPIS relies on a set of repair heuristics to recover from inconsistencies [7], the macro-opportunistic scheduler of this study was built to use the same consistency enforcing techniques and the same backtracking scheme as MICRO-BOSS[4]. 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 [9].

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 [9] for further details).

Figure 3-1 and 3-2 summarize the results of the comparison between MICRO-BOSS and the macro-opportunistic scheduler[5]. The macro-opportunistic scheduler was consistently outperformed by MICRO-BOSS (under all eight scheduling conditions) both with respect to tardiness, flowtime (i.e. work-in-process) and in-system time (i.e. total inventory, including finished-goods inventory). These results also 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.

[3]See [9] for further details.

[4]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.

[5]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.

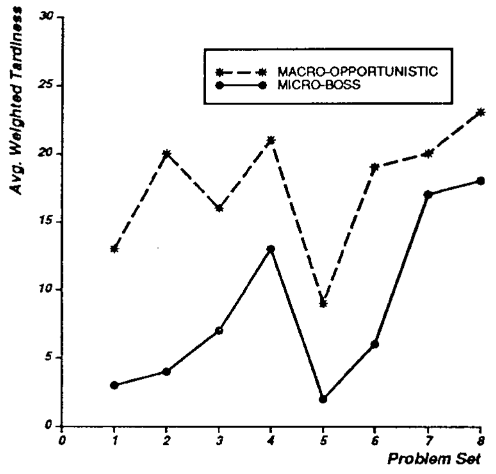

Figure 3-1: Tardiness performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

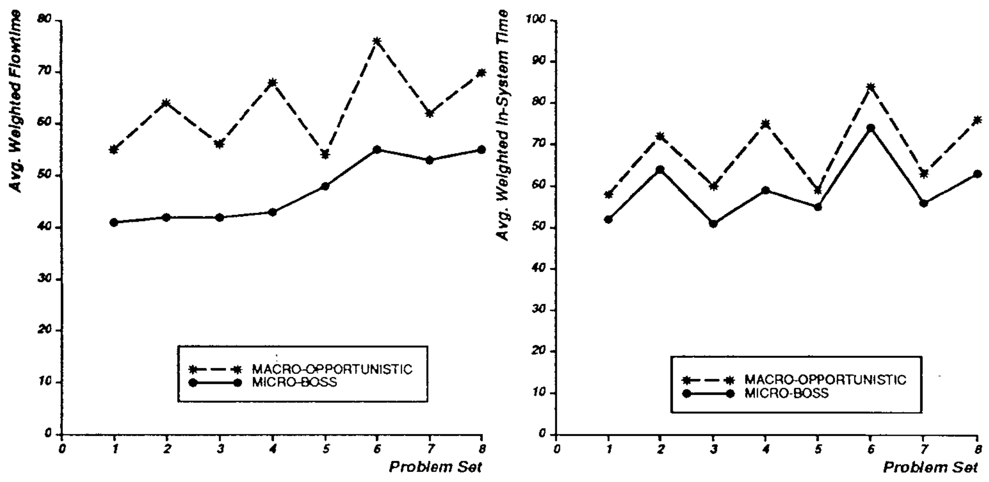

Figure 3-2: Flowtime and in-system time performance of MICRO-BOSS and the macro-opportunistic scheduler on eight different problem sets.

Additional experimental results are also reported in [9], including a comparison against six different priority dispatch rules coupled with different release policies, and an empirical evaluation of the impact of using biased demand profiles to identify important tradeoff operations.

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 [6], 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.

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. R.E. Fox. OPT: Leapfrogging the Japanese. In *Just-in-time Manufacture*, IFS Ltd, Springer Verlag, 1987.

3. S. French. *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop.* Wiley, 1982.

4. 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.

5. T.E. Morton, S.R. Lawrence, S. Rajagopolan, S Kekre. "SCHED-STAR: A Price-Based Shop Scheduling Module". *Journal of Manufacturing and Operations Management* (1988), 131-181.

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

7. 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.

8. Peng Si Ow and Thomas Morton. "The Single Machine Early/Tardy Problem". *Management Science 35*, 2 (1989), 177-191.

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

10. 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.