Online Scheduling for Reprographic Machines

Markus Fromherz
Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304
fromherz@parc.xerox.com

Lise Getoor
CS Dept
Stanford University
Stanford, CA 94305-9010
getoor@cs.stanford.edu

Introduction

We present a real-world online scheduling application. In this application, the problem input is fed incrementally to the scheduler, and the scheduler has only a portion of the entire job available before it has to start making scheduling decisions. In fact, job submission, scheduling and execution may all happen in parallel and at different speeds.

We are investigating constraint-based online scheduling in the domain of reprographic machines (photo-copiers, printers, and fax machines). The task of a generic reprographic machine scheduler is to schedule the operations that produce a given document in real-time. In particular, we focus on optimizing the make-span of the schedule, i.e., minimizing the output time of the last sheet. The make-span is usually taken as a measure of a machine's productivity. For high-end machines, productivity improvement often translates proportionally to an increase in perceived value. For expensive machines, even small improvements (e.g., by 5-10%) are significant.

Constraint programming (Van Hentenryck 1989) and the large body of experience with constraint-based scheduling (Zweben & Fox 1994) provide a well-suited foundation for online scheduling. However, optimizing overall productivity is difficult in an online environment, and even more so with a real-time algorithm. Not only is it difficult to choose the right schedule without complete job information, but there may not even be enough time to find that schedule. This has motivated our work on heuristics for online scheduling (Getoor *et al.* 1997). Here, we point out further research directions in this area.

Reprographic Machine Scheduling

The task of scheduling networked reprographic machines is defined as follows. A scheduler is given a *machine description* and a finite *sequence of output sheets* that are to be printed by the machine. The machine description is formulated in terms of constraints on sheets. The scheduler's task is to determine when the operations necessary to feed and print sheets have to be performed such that the output sequence is produced. In networked machines (e.g., printers), the entire document is available up front and disclosed to the scheduler at a faster rate than it is printed, i.e., the scheduler usually is able to look ahead and take into account part or all of the job when scheduling each sheet.

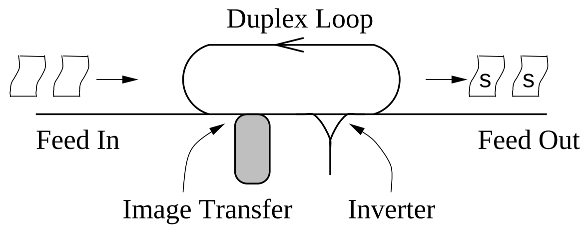

Figure 1: Schematic view of a simple reprographic machine

For this paper, we assume that the machine has one input and one output (Figure 1). In between, sheets are continuously moving, and once a sheet is fed into the machine, its itinerary through the machine is fixed. Thus, *the scheduler's task is to determine the timed sheet input sequence for a given output sequence.* In the following, we will discuss a simplified but complete form of the problem of scheduling reprographic machines. While high-end machines have more complex constraints, supporting color, buffering and multiple sheet sizes, this simpler version already represents a realistic and non-trivial machine.

Basic Machine Model

In this model of a simple machine (Figure 1), we are given a configuration with an image transfer component, an inverter component, and a paper path that leads from the input to the image transfer, the inverter component, and then either out or back to the image transfer. A single-sided sheet is printed once

Document O	`sddd`
Naive S	`sf___bf___bf___b`
Greedy S	`sfff_bbb`
Optimal S	`fff_sbbb`

Figure 2: Alternative schedules S for the same document O

and moved to the output without inversion. A double-sided sheet is printed on one side, then inverted and moved back to be printed on the other side, before it is inverted again and moved to the output.

The transportation and printing of sheets and images is constrained in various ways by the physics of the machine. For example, for a machine to operate properly, sheets cannot overlap in the paper path; sheets and images have to be synchronized; and images can be placed on the photo-receptor belt only at certain places (because a seam in the belt must be avoided). Inversion takes longer than bypassing; therefore inverted sheets cannot immediately be followed by non-inverted sheets at the entrance of the inverter. These are the constraints to be honored when scheduling an output sequence on the machine. (See (Getoor *et al.* 1997) for more details.)

Figure 2 shows different schedules that could be generated for a given job, using either a naive, greedy, or optimal (look-ahead) algorithm. This figure shows different arrangements for placing patterns for a single-sided (`s`) or double-sided (`f___b`) sheet. The blanks `_` between front side `f` and back side `b` of a double-sided sheet represent times where other images could be printed; the distance between `f` and `b` corresponds to the time it takes the sheet to travel around the duplex loop. Note however that if we add sheets to our desired output, $O =$ `sddddd`, then the optimal schedule becomes $S =$ `sfffffbbbbb`, i.e. the additional sheets change our scheduling decisions.

Properties of the Basic Model

One of the properties that makes scheduling for certain machine models in this domain tractable is that individual sheets in a job have limited extent: because of the strong output ordering requirements of this domain, and the fixed paper path per sheet type in the basic models, there are a limited number of plausible timings for the sheets.

This is a very desirable domain property that allows us to impose quite strong local constraints on the placement of individual sheets in the job in the schedule. Note, however, that a scheduling decision for a sheet does impact the scheduling decisions for sheets arbitrarily far away from it in the input; thus there is no simple greedy algorithm that is guaranteed to find the optimal solution (Fromherz & Carlson 1994). However, scheduling for the basic machine is not NP-hard; there is a pseudo-polynomial time algorithm for finding the optimal schedule (Fromherz & Carlson 1994).

Research Directions for Online Scheduling

As mentioned, it is usually not possible to wait until the scheduler has received a complete job and generated a schedule for it. In addition, scheduling and schedule execution typically happen in parallel. At discrete time instants (e.g., once a second with the above machine model), the scheduler is interrupted and asked if a sheet is to be fed (and which). The scheduler typically gets a few tenths of that time for its processing.

Thus, an online scheduler often has to schedule sheets without knowing the full job. As explained, optimal schedules for a partial job do not necessarily extend to optimal schedules for the entire job. Thus, the question is: which online decision strategy leads to overall optimal productivity?

In this context, experiments on the basic model have shown that a good strategy leading to overall optimal productivity for the type of machine shown above is *full optimization with minimal commitment* (Fromherz & Carlson 1994): whenever the scheduler is interrupted, generate an optimal schedule for the job known at that time, but commit only to that part of the schedule that has to be returned for execution. We found that a competitive ratio of 1.05 between an online algorithm following this strategy and an offline algorithm generating optimal schedules for randomly generated jobs is typical. This is consistent with an earlier theoretical analysis of the basic model (Motwani, Saraswat, & Tong 1993). These ideas have to be further explored for more complex machine models.

In addition, we can also examine the use of such online strategies for offline algorithms. Figure 3 shows the overall run times of offline and online algorithms for the same jobs. Even though the online algorithm is repeatedly generating optimal schedules for larger and larger parts of the job, this algorithm scales much better with job length than the offline algorithm. Given that the quality of this online algorithm is within reasonable bounds (in this case, the online schedule was always within 1% of optimal), this suggests that it may be beneficial in certain domains to treat offline problems artificially as online problems. In other words, even if the entire job is known, generating partial schedules incrementally may provide a good trade-off between efficiency and optimality. The generality of this finding

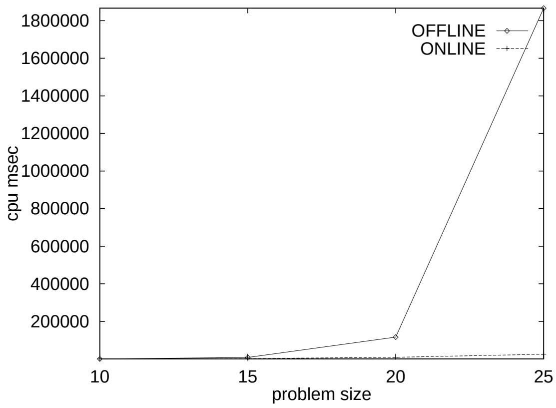

Figure 3: Total run times of online vs. offline algorithms, averaged over 25 runs of random jobs of length 10, 15, 20 and 25.

has to be validated in further research.

References

Fromherz, M., and Carlson, B. 1994. Optimal incremental and anytime scheduling. In Lim, P., and Jourdan, J., eds., *Proc. Workshop on Constraint Languages/Systems and their Use in Problem Modeling at ILPS'94*, 45–59. TR 94-38.

Getoor, L.; Ottosson, G.; Fromherz, M.; and Carlson, B. 1997. Effective redundant constraints for online scheduling. In *Proc. AAAI'97*.

Motwani, R.; Saraswat, V.; and Tong, E. 1993. Online printer scheduling. Xerox PARC, Research Note.

Van Hentenryck, P. 1989. *Constraint Satisfaction in Logic Programming*. Logic Programming Series. Cambridge, MA: MIT Press.

Zweben, M., and Fox, M. S., eds. 1994. *Intelligent Scheduling*. San Francisco, Calif.: MK.