Lookahead Scheduling of Unrelated Machines

Ben Coleman and Weizhen Mao
Department of Computer Science, College of William & Mary, Williamsburg, VA 23187-8795, USA
{*coleman,wm*}*@cs.wm.edu*

Abstract

We consider the unrelated machine scheduling problem where each job may have a different processing time on each machine. This model accurately represents systems where each machine or worker has different abilities. A special case of this problem is truck scheduling in a cross-docking environment where incoming trucks are scheduled to doors and their cargo is moved to outgoing trucks. We study two algorithms that attempt to minimize the average wait time for the general unrelated machine scheduling problem, a non-lookahead algorithm and a lookahead algorithm. We show by extensive simulation that the lookahead algorithm produces schedules that are up to 35 percent better than the non-lookahead model.

Keywords: Job Scheduling, Cross-Docking, Heuristic, Simulation, Lookahead

1 Problem Description

We are interested in job scheduling of unrelated machines, where each machine has different abilities. Formally, we have a set of n jobs we wish to schedule on m machines. In addition to an arrival time r_j, each job J_j has a set of processing times $p_{i,j}$ representing the time requirement on machine i, for $i = 1, \ldots, m$. We are interested in scheduling algorithms that minimize the average wait time, $\frac{1}{n} sum_{j=1}^{n} w_j$, where w_j is the amount of time between when a job arrives, r_j, and when it begins execution.

Unrelated machine scheduling is useful when modeling systems where processing times are determined based on which machine executes each job. For example, a special case of the unrelated machine scheduling problem is cross-docking [3]. Here, we have a set of receiving doors where trucks are unloaded and a set of shipping doors where cargo is loaded. Once a truck is assigned to a receiving door, the pallets on that truck must be unloaded and taken to the appropriate shipping door. See Figure 1 for a representative cross-docking layout: $R_1, \ldots R_r$ are receiving doors and $S_1, \ldots S_s$ are shipping doors. In this system, consider a truck full of pallets destined for various shipping doors. Since each pallet must be moved to the appropriate shipping door, the choice of receiving door determines the unloading time of the truck.

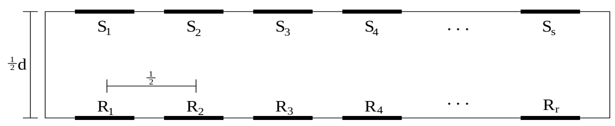

Figure 1

Unlike the general unrelated model where processing times can be arbitrarily large or small, the unloading times for any truck is bounded by the size of the cross-dock and the number of pallets on the truck. Assume we have a truck of c pallets all destined for shipping door S_1. If the truck is assigned to receiving door R_1, then the unloading time is minimized at $c \cdot d$. Likewise, if the truck is assigned receiving door R_r, the maximum time of $c \cdot (r + d)$ is needed.

In this paper, we describe a realistic model for unrelated job scheduling and then describe two algorithms for making scheduling decisions in this model. After explaining why analytical study is untenable, we compare the algorithms using simulation. This simulation demonstrates that the lookahead algorithm produces schedules with average wait times that are up wards of 35 percent better than the non-lookahead algorithm.

Although many previous scheduling results use lookahead [5, 7], the definitions differ. Our work is a continuation of Mao and Kincaid [6], who show the benefits of lookahead on a one-processor scheduling problem. We have also shown the benefit of lookahead in a two identical processor system [2]. Lookahead has not been applied to the general unrelated machine model. Gue [3] considered lookahead in the cross-docking model, however his definition simply means knowledge of the cargo of the truck.

2 Lookahead Model

A scheduling algorithm must decide when to execute each job and on which machine. An offline algorithm knows everything about the jobs when making decisions. It knows when jobs will arrive and the processing times of each job. Using this information, an offline algorithm can create an optimal schedule. On the other hand, an online algorithm considers the jobs in the order of arrival and schedules each job without knowledge of future jobs.

We are interested in algorithms that are different from the conventional offline and online algorithms in two ways. First, we utilize lookahead. Unlike an offline algorithm that knows everything about the future jobs or an online algorithm which considers only the current job, a lookahead algorithm has knowledge of a portion of the jobs that will arrive in the near future. We feel that this middle ground represents a more realistic situation. Consider a doctor who responds to patients' requests for an office visit. The doctor is unable to know all such requests that will occur in the future, however, she has access to an appointment book which records all requests in the near future, say, up to next month. In general, lookahead is used to find out future requests and then allocate resources in such a way to optimize the quality of service.

The second difference relates to the algorithm's sense of time. In effect, offline and online algorithms operate outside time. Both algorithms are free to schedule jobs at any time, so long as it is not earlier than the arrival time of the job. In our model, all decisions are made relative to flowing time. If a job is not immediately scheduled when it arrives, it is place into a wait queue.

Because of these differences, our lookahead algorithm represents a more realistic approach that falls between overly optimistic offline algorithms and pessimistic online algorithms. To implement a lookahead algorithm, we need a variety of components. First, we require a wait queue to hold jobs that have arrived but are not yet scheduled. To incorporate lookahead, we need a second queue to hold information about the jobs that will arrive in the near future. We also need two processes, one to maintain the queues by adding and deleting jobs, and another to make scheduling decisions, using the algorithms we have developed.

3 Algorithms

We are interested in the improvement of a lookahead algorithm over a traditional online algorithm. Both algorithms make decisions relative to flowing time, the only possible decisions are to schedule a job on an idle processor or to wait. Since neither algorithm has complete knowledge of the future, neither can make the optimal decision in all cases. The best each algorithm can do is to make a decision using the known information. Specifically, the algorithm assumes that no further jobs will arrive and decides to wait or schedule based solely on the known jobs.

For the non-lookahead algorithm, assuming that no further jobs will arrive reduces the problem to the offline problem where all jobs are known in advance. Consequently, the non-lookahead algorithm never decides to hold a machine idle, as doing so only increases the wait time of jobs. The decision the non-lookahead algorithm makes determines which waiting job will be scheduled on the idle processor. The solution to the offline problem can be found by translating the problem into a weighted matching problem [1].

We compare this online algorithm with our lookahead-1 algorithm. In addition to knowledge of jobs in the wait queue, this algorithm also has access to the lookahead queue, containing the next job to arrive. Using this information, the lookahead-1 algorithm can make scheduling decisions that create opportunities for improvement in the overall schedule. Specifically, by causing a small additional wait of an already waiting job, the lookahead algorithm can greatly reduce the wait of the lookahead job and consequently reduce the average wait time of the entire sequence.

Similar to the non-lookahead algorithm, the lookahead algorithm assumes that no further jobs will arrive other than the lookahead job. Next, the lookahead algorithm computes the non-waiting schedule where no machine is ever held idle. In this schedule, the lookahead job is simply added to the wait queue when it arrives. The lookahead algorithm compares this schedule with the waiting schedule, where a machine is held idle until the lookahead job arrives and is executed. If the waiting schedule produces a better schedule, the algorithm will wait, otherwise it will execute a job according to the non-waiting schedule.

4 Algorithm Analysis

Intuitively, the lookahead algorithm should produce better schedules than the non-lookahead algorithm, and we would like to quantify this improvement. Competitive analysis is the traditional method to analyze online algorithms and can be used to study lookahead algorithms. With this technique, we compare an algorithm to the idealized, optimal algorithm that produces the best schedule in all instances. We say that an algorithm A is c-competitive if, for all instances, $A(I) \leq c \cdot OPT(I)$ where $A(I)$ is the average wait time for instance I given by A and $OPT(I)$ is the optimal.

For unrelated machine scheduling with job arrival times, it has been shown that there cannot exist an algorithm with a bounded competitive ratio [4]. That is, for any algorithm, there exists an instance where the optimal algorithm produces a schedule that is infinitely better. The basis for this infinite ratio is that after any job is scheduled on machine i, it is possible that another job arrives that has massive processing time on all machines except machine i. Since that machine is already busy, the algorithm ends up executing the job using a large processing time. The optimal schedule "knows" which machines to leave idle and avoids the large processing times.

Since we know that we cannot create an algorithm with bounded competitive ratio, we are interested in other methods to evaluate algorithms for this problem. In this case, the use of simulation is appropriate.

5 Simulation Results

One of the most important properties when setting up a simulation is stability. If jobs arrive faster than they can be processed, the number of jobs in the wait queue will grow toward infinity. Stability is achieved as long as the jobs arrive at a rate less than or equal to the rate they can be processed. We are interested in the behavior of the system at the stability point.

In a traditional job scheduling model, where each job has the same processing time on each machine, the stability point occurs when the average inter-arrival time equal the average processing time divided by the number of machines. In our model, each job has a range of possible processing times, but only one time will actually be utilized. When the machines can process the arriving jobs faster than the arrival rate, stability is not an issue and it does not matter which time is used. At the stability point, the most efficient way to process jobs is to schedule each job where it requires the least processing time.

Based on this observation, given the distribution for processing times, we can calculate the appropriate arrival rate to achieve stability. The average processing time at the stability point will be the minimum of m processing times. This value can be modeled as a Markov chain and computed in closed form when processing times are drawn from standard distributions.

Intuitively, lookahead will be most beneficial when a large job is held to wait for the arrival for a small job. To test this idea, we simulated a 2 machine environment of unrelated machines. Inter-arrival rates were assumed to be exponentially distributed and service times were drawn from a variety of hyper-exponential distributions. This choice was made because it allowed us to compute the stability point in closed form and because it allowed a high variance of processing times. In general, this means that very long jobs will occur along with small jobs.

When running the simulation, all parameters except the processing times and arrival rates were fixed. For 100 jobs, we varied the processing times and then used the closed-form solution of the Markov chain to calculate the corresponding arrival rate to achieve stability. Specifically, we drew from a hyper-exponential(μ_1, μ_2, α) with μ_1 fixed at 1, μ_2 varying from 5 to 95, and α ranging from .5 to .85. Figure 2 summaries the improvement in average wait time of the lookahead algorithm over the non-lookahead algorithm.

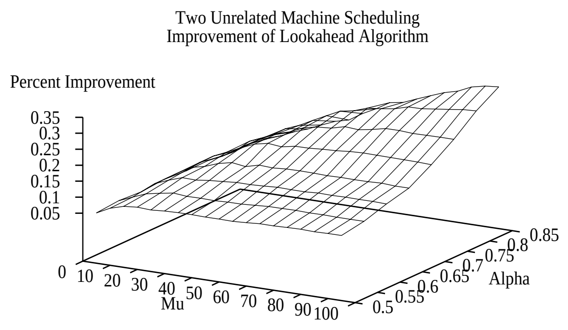

Figure 2

The lookahead algorithm produces schedules that are between 10 and 35 percent better than the corresponding non-lookahead schedule. In part, this improvement corresponds to an increase in the variance. The variance of a hyper-exponential distribution is defined to be

$$\alpha \cdot \mu_1 + (1 - alpha) \cdot \mu_2$$

As μ_2 increases, the variance of the processing times also increases. By creating jobs with larger processing times, we create opportunities for the algorithm to hold a machine idle to wait for a smaller job to arrive.

However, simply increasing the variance does not mean the lookahead algorithm will have better performance. As α increases, the variance of the hyper-exponential distribution decreases, especially as μ_2 became larger. In this case, when a large job is in the wait queue, it is much more likely that the lookahead will be a small job and the algorithm can gain improvement by waiting. Therefore, improvement occurs because long jobs are delayed for the arrival of short jobs.

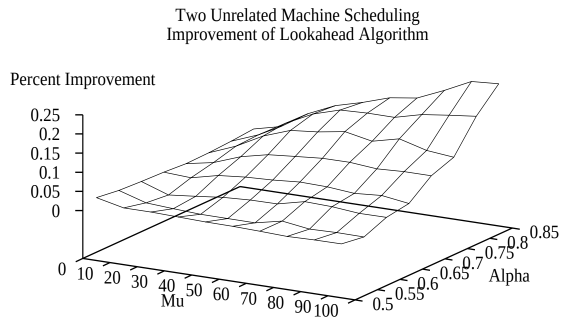

Figure 3

The use of 100 jobs demonstrates the transient behavior of the system. We also ran the simulation with the 1000 jobs. Figure 3 summarize these results. For small values of α, the distribution of jobs gives numerous large jobs with a slight majority of small jobs. In these situations, the lookahead algorithm produces little improvement. However, for large values of α, the large majority of the jobs are short and up to 30 percent improvement can still be made. These observations further support the idea that when long jobs are delayed to wait for the arrival of short jobs, improvement is made. In general, the strength of lookahead is this ability to know when it is beneficial to hold a machine idle.

6 Future Work

Although competitive analysis cannot be applied in a meaningful way to unrelated machine scheduling, we believe it can be used in the cross-docking model. For any truck, the unloading time is at least the number of pallets times the distance to the closest shipping door. Likewise, the maximum unloading time is based on the distance the the furthest shipping door. Given these upper and lower bounds of processing time for any truck, we hope to find a meaningful competitive ratio for this model.

References

[1] J. Bruno, Jr. E. G. Coffman, and R. Sethi. Scheduling independent tasks to reduce mean finishing time. *Communications of the ACM*, 17(7):382–387, 1974.

[2] B. Coleman and W. Mao. Lookahead scheduling in a real-time context. In *Proceedings of the 6th International Conference on Computer Science a nd Informatics*, 2002.

[3] K. R. Gue. The effects of trailer scheduling on the layout of freight terminals. *Transportation Science*, 33(4):419–428, Nov 1999.

[4] Han Hoogeveen, Petra Schuurman, and Gerhard J. Woeginger. Non-approximability results for scheduling problems with minsum criteria. *Lecture Notes in Computer Science*, 1412:353–366, 1998.

[5] P. Keskinocak. Online algorithms with lookahead: A survey. ISYE Working Paper, 1999.

[6] W. Mao and R. K. Kincaid. A look-ahead heuristic for scheduling jobs with release dates on a single machine. *Computers and Operations Research*, 21(10):1041–1050, 1994.

[7] S. Phillips and J. Westbrook. On-line algorithms: Competitive analysis and beyond. In M. J. Atallah, editor, *Algorithms and Theory of Computation Handbook*, chapter 10. CRC Press, Boca Raton, 1999.