Dispatching Rules

Abdullah Karaman [*]
Department of Industrial Engineering
Bilkent University
TR-06533 ANKARA
akaraman@bilkent.edu.tr

June 16, 2000

Abstract

Most scheduling problems are *NP-hard*. For such problems, there is strong evidence that it is highly unlikely to find an algorithm that runs in polynomial time. For this reason, enumerative techniques are used to find optimal solution of a scheduling problem, but the computational effort needed by these techniques grows very fast as the size of the problem increases. So, in solving these scheduling problems some heuristics are used to attain reasonably good solutions in relatively short time. Dispatching rules are general purpose heuristics that have proven to be useful in scheduling theory. These rules do not guarantee an optimal solution but they provide near optimal solutions in a reasonable amount of time.

1 Introduction

In the words of Pinedo and Chao [1], "dispatching rule is a rule that prioritizes all the jobs that are waiting for processing on a machine." In other words, it allows an idle machine to select its next operation from among currently available waiting jobs at the corresponding machine. It defines a specific sequence decision each time a machine gets idle. Rather than determine a global optimal sequencing policy, it provides a heuristic optimum among alternative sequencing strategies. The selection of job is made by considering the properties of individual jobs, properties of machines and current time. A priority function assigns a priority value to each job based on the above criteria and selects the job with the highest priority.

[*]In partial fulfillment of the requirements for the course IE 572 Spring 2000

In the scheduling literature, the terms scheduling rule, dispatching rule and heuristic are often used synonymously. Gere [2] has made an attempt to distinguish between these. He suggested a distinction between these terms and considers dispatching rules as simply as a technique by which a number is assigned to each job in the queue. He defines a heuristic to represent the rules of thumb and scheduling rule a combination of these dispatching rules and heuristics [3].

Dispatching rules can be classified whether they are time dependent or not. If the priority value of a job does not change with time, or stating differently, it is time-independent, then these rules called *static rules*. On the other hand, *dynamic rules* are time-dependent. They make use of the information on the current state of the schedule. These rules involve changing the priority values as the time goes on. The distinction between static and dynamic rules is due to Jackson [4].

Another way of classifying dispatching rules based on the information they require. Conway and Maxwell [5] describe *local dispatching rules* as those that require information only about jobs that are waiting at a machine, while *global dispatching rules* require additional information about jobs or machine states at other machines [3].

In a paper by Panwalkar [6], he divided the literature of scheduling rules into two periods. First period begins with the pioneering paper of Johnson [7] (Johnson provided an analytical study of a scheduling problem) and ends in 1975. In this period analytical methods of static problems considered and optimal algorithms were developed such as branch and bound and dynamic programming. Also, some simulation studies in dynamic environment considered. A thorough paper that reviews the research of this period is Panwalkar and Iskander [8]. In this paper, over 110 scheduling rules and 36 papers about this period is discussed. The second period covers the years since then. In this period the theory grew faster. The computational complexity of such problems and worst case performance of heuristics are among the main research areas.

2 Basic Dispatching Rules

There are many dispatching rules. In this section a small sample of these rules and the environments in which they are used are presented. The use of these rules is based on the machine environment and the objective function of the problem. For some objective function, such as total flow time, an optimal solution can be obtained by a procedure as simply as sorting the jobs. On the other hand, for some other objective functions, such as total weighted tardiness, no simple solution procedure is available, and combinatorial optimization techniques is used.

- Shortest Processing Time Rule (SPT) The shortest processing time rule orders the jobs in the order of increasing processing times. Whenever a machine is freed, the shortest job ready at the time will begin processing. This algorithm is optimal for finding the minimum total completion time and weighted completion time. In the single machine environment with ready time at 0 for all jobs, this heuristic is optimal in minimizing the mean number of jobs in the system, maximum waiting time and mean lateness.

- Earliest Due Date Rule (EDD) The earliest due date rule orders the sequence of jobs to be done from the job with the earliest due date to the job with the latest due date. In a single machine environment, *EDD* rule is optimal when one wants to minimize the maximum lateness, or to minimize the maximum tardiness.

- Weighted Shortest Processing Time (WSPT) This is a variation of *SPT* rule. Let $p(i)$ and $w(i)$ denote the processing time and the weight associated with the $i'th$ job to be done. Then *WSPT* orders the jobs such that the following inequality holds,

  $$\frac{p(1)}{w(1)} \leq \frac{p(2)}{w(2)} \leq \ldots \leq \frac{p(n)}{w(n)}.$$

  In a single machine environment, The *WSPT* rule minimizes the weighted sum of the completion time. When all weights are equal, then *WSPT* reduces to *SPT*.

- Minimum Slack (MS) When a machine is freed at time t, the slack of a job is defined by the identity $\max d_j - p_j - t$. *MS* schedules the job with the minimum slack next. In a single machine environment with ready time at 0, *MS* minimizes the maximum lateness.

- Shortest Setup Time (SST) Whenever a machine is freed, this rule selects the job with shortest setup time.

- Longest Processing Time (LPT) The longest processing time rule orders the jobs in the order of deceasing processing times. Whenever a machine is freed, the largest job ready at the time will begin processing. This rule is used when the objective function of the problem is to find minimum makespan of the schedule.

- Critical Ratio (CR) The critical ratio defined as $\frac{d_j - t}{p_j}$. Jobs are scheduled in ascending order of the critical ratio.

2.1 Discussion

Among the rules discussed above, the *EDD* rule is a static rule and the *MS* rule is a dynamic one. *EDD* rule is static since the priority value of jobs are not time dependent. However, *MS* rule is dynamic because at time t, job j may have a higher priority value than job k, but at a later time job k can have a higher priority value. The value needs updating as time progresses.

A comprehensive listing and definitions of various rules investigated up to mid-1970s can be found in a survey paper by Panwalkar and Iskander [8]. Haupt [9] did a survey on priority rule based job shop scheduling. Hoffmann and Scudder [10] provides a concise summary of these rules.

These rules described here will not give a good solution when a complex objective function has to be minimized. In the next section, the composite dispatching rules which are combination of basic dispatching rules will be discussed.

3 Composite Dispatching rules

One of the questions that researchers try to find an answer to is when and how priority rules are most effective. In order to find an answer, these rules studied within a variety of machine environments. When objective function is complicated, in other words it is a combination of several basic objective functions, these basic dispatching rules did not help much. In fact, most of the real world problems have multiple objectives. Hence, some combination of basic dispatching rules need to be considered.

As stated in Pinedo and Chao, "a composite dispatching rule is a ranking expression that combines a number of elementary dispatching rules [1]." It is combination of basic rules with associated weights. Each basic rule affects the priority value of the job to some extent. This is determined by a scaling parameter and determined by the designer of the rule. This scaling parameter can be computed by simulation or some other statistical methods. So, whenever a composite dispatching rule will be used, the necessary statistics are computed and used in the calculation of scaling parameter of basic dispatching rules.

One example of the composite dispatching rule is *apparent tardiness cost (ATC)* heuristic. This rule is used in a single machine environment and n jobs (all available at time 0). The objective is to minimize sum of the weighted tardiness. The computational effort needed by combinatorial methods avoids the usage of these methods. Some basic dispatching rules that fits into this problem were considered. The resulting rule is the well known *apparent tardiness cost (ATC)* rule. The *ATC* rule is a combination of *WSPT* and *MS* (recall that *MS* tends to minimize due date related objectives and *WSPT* tends to minimize weighted sum of the completion times). The priority value of jobs can be calculated by the identity

$$I_j(t) = \frac{w_j}{p_j} \exp\left(-\frac{\max(d_j - p_j - t, 0)}{K\overline{p}}\right)$$

where p_j corresponds to processing time, w_j corresponds to weight, t denotes time, d_j due date and $\overline{p}$ is the average of the processing times of the remaining jobs. Here k denotes the scaling parameter and can be determined empirically. The contribution of basic rules can be determined by changing the K value. In this example, if K is very large, then this rule reduces to *WSPT* and if K is very small it reduces to *MS*. The K value can be changed according to the problem instance. For further information and an example of *ATC* rule can be found in [1]

4 Conclusion

In solving a scheduling problem to optimality, computational effort required by combinatorial methods grows remarkably fast as the size of the problem increases. So, some heuristic procedures have been proved useful in solving these problems to reasonable good solutions in a relatively short time. Dispatching rules are useful for finding a good solution with regard to a single objective function.

References

[1] M. Pinedo, X. Chao, *Operations Scheduling with Applications in Manufacturing and Services*, McGraw Hill, 1999.

[2] W.S.Gere, Heuristics in job shop scheduling, *Management Science* 13:167-190, 1966.

[3] R.Ramasesh, Dynamic job shop scheduling: A Survey of operations research, *Omega International J. of Mgmt Sci.*, 18:43-57, 1990.

[4] J.R. Jackson, Simulation research on job shop scheduling, *Naval Research Logistics Quarterly*, 4:287-295, 1957.

[5] R.W. Conway, W.L. Maxwell, Network dispatching by the shortest operation discipline, *Operations Research*, 10:51-73, 1962.

[6] S.S. Panwalkar, Scheduling rules revisited, *Proceedings of the Joint National Meeting of Scheduling*, 1990.

[7] S.M. Johnson, Optimal two and three stage production schedules with setup times included, *Naval Research Logistics Quarterly*, 1:61-68, 1954.

[8] S.S. Panwalkar, W. Iskander, A Survey of scheduling rules, *Operations Research*, 25: 45-61, 1977.

[9] R. Haupt, A Survey of priority rule-based scheduling, *OR Spektrum*, 11:3-16, 1989.

[10] T.R. Hoffmann, G.D. Scudder, Priority scheduling with cost considerations, *International Journal of Production Research*, 21:881-887, 1983.