Research Notes for Chapter 2[*]

Scientific developments can always be made logical and rational with sufficient hindsight. It is amazing, however, how clouded the crystal ball looks beforehand. We all wear such intellectual blinders and make such inexplicable blunders that it is amazing that any progress is made at all. (Richard Bellman)

Many of the basic models we covered in this chapter appeared for the first time in a seminal paper by Smith (1956). His core result was a characterization of cases where simple transitive sorting rules yield optimal sequences. He then proceeded to cover some examples, including SPT, SWPT and EDD, not to mention Smith's rule, which we cited in the main text. Smith's paper is deceptively simple. In hindsight it may be difficult to comprehend how these simple and important sorting rules were not recognized before. But hindsight is sometimes very hazy when it comes to distinguishing between the trivial and the elegant. In fact, no single scheduling paper before or after Smith (1956) solved so many important instances and provided so much insight into the essence of the difficulty of other instances. We should note that the EDD result is often attributed to Jackson (1955), which is an unpublished working paper. According to Smith, however, Jackson had generalized EDD to the dynamic arrival case, which suggests that the basic EDD is still due to Smith. One should note, in general, that the publication process is often measured in years, so a 1956 publication can easily be about a result that had been developed before 1955. Thus, just because the Jackson paper was released in 1955 does not imply that it is earlier than Smith's work. Finally, the terms SPT, SWPT and EDD are more recent. For example, to our knowledge, the term EDD was coined by Emmons (1969). Furthermore, most sources, including Baker (1974), refer to SWPT as WSPT. The reason for the switch to SWPT—introduced in early versions of Baker (2005)—is because it describes the solution process better: first, by dividing each p_j by w_j, we obtain weighted processing times (WPT); only then is the list sorted from smallest to largest (SWPT). In contrast, WSPT connotes applying weights to the jobs after they are already in SPT sequence.

An important motivation for careful study of basic theorems is improving our intuition. When viewed in this light, Theorem 2.7 stands out, because the slack rule is one of the most prevalent intuitive rules used by practitioners, and it can be quite a challenge to demonstrate that it can yield undesired results. Furthermore, the slack rule had been promoted by many published papers, mostly in the context of project scheduling. In that context, at least for a single project, the main concern is with maximal tardiness so the slack rule is less damaging than in machine scheduling, where it essentially increases total tardiness unnecessarily. Practically without exception, these published results ignored Theorem 2.7. Nonetheless, the earliest source of this theorem is Conway, Maxwell and Miller (1967), and as such it predates those papers. We discuss such rules in the context of projects in Chapter 17.

[*] The Research Notes series (copyright © 2009, 2019 by Kenneth R. Baker and Dan Trietsch) accompanies our textbook *Principles of Sequencing and Scheduling*, Wiley (2009, 2019). The main purposes of the Research Notes series are to provide historical details about the development of sequencing and scheduling theory, expand the book's coverage for advanced readers, provide links to other relevant research, and identify important challenges and emerging research areas. Our coverage may be updated on an ongoing basis. We invite comments and corrections.

Citation details: Baker, K.R. and D. Trietsch (2019) Research Notes for Chapter 2 in *Principles of Sequencing and Scheduling* (Wiley, 2019). URL: http://faculty.tuck.dartmouth.edu/principles-sequencing-scheduling/.

Algorithm 2.1 was published by Moore (1968). Moore introduced a slightly complex procedure and then presented a streamlined version—our Algorithm 2.1—which he attributed to Thom Hodgson. For this reason the literature often refers to Algorithm 2.1 as the "Moore-Hodgson Algorithm" or the "Hodgson Algorithm." In Chapter 7 we discuss a stochastic generalization of this approach—Algorithm 7.1—that relates more closely to the original version of Moore, but in the deterministic context Algorithm 2.1 is more efficient. In more detail, the original Moore algorithm sorts jobs by SPT order and tentatively appends them to B, one at a time. The jobs in B are then tested for feasibility by putting them in EDD sequence and checking if any of them is tardy. If the tentative job causes infeasibility, it is moved from B to A. To put the jobs in B into EDD order, it is sufficient to insert the last, tentative, job into its correct position in the previous subsequence, which takes $O(\log n)$ comparisons per job (by bisection search). However, checking whether this insertion causes tardiness downstream takes $O(n)$ per job, which dominates the complexity of the insertion step. Although jobs are considered in a different order, the net result is identical to that of the Moore-Hodgson version, and the complexity is $O(n^2)$. The Moore-Hodgson version is more efficient because the feasibility test for job j only requires adding p_j to the previous completion time of earlier jobs in B and comparing the result to d_j, which is $O(1)$ per job, and thus $O(n)$ throughout. If there are few or no tardy jobs then the savings can be considerable. According to numerous sources in the literature, the computational complexity of the streamlined version is $O(n \log n)$. Moore himself states that his algorithm requires sorting—which takes $O(n \log n)$—plus $n(n + 1)/2$ additions and comparisons. When discussing the streamlined version he comments that it saves up to $n(n + 1)/2$ additions and comparisons. This, however, does not imply that it always saves *all* the $n(n + 1)/2$ additions and comparisons. Indeed, a straightforward application of Algorithm 2.1 is also $O(n^2)$. To see this consider that upon tardiness, which may occur $O(n)$ times, we must find the largest job to remove from B to A. Finding that job requires a search that consumes $O(n)$ operations. However, whereas a straightforward application takes $O(n^2)$, it is indeed possible to code the algorithm to take $O(n \log n)$ time. To this end we store the B set in a binary max heap. A binary max heap is a binary tree data structure where elements are not fully sorted and yet the maximum is maintained at the root of the tree, the second largest item is at the next level and so on. It takes $O(\log n)$ to add an element to a binary max tree while maintaining this property, or to delete the root and rearrange the heap. Therefore, every time we add an element to B, we also store it in the binary max heap, at $O(\log n)$ time. We have to do that n times, so the total is $O(n \log n)$. When we have to delete a job, we delete the root and rearrange the heap such that the new maximum is at the root again. As each such step takes $O(\log n)$, the total complexity of deletions cannot exceed $O(n \log n)$.[†]

Another result published in Moore (1968) and attributed by him to Eugene Lawler was later eclipsed by a well known short note due to Lawler himself in 1973. This note provides a very simple-looking algorithm for a slightly more general problem, which can also accommodate precedence constraints. We present the basic Lawler (1973) result in the next chapter as Theorem 3.1 (and we extend it to accommodate precedence constraints in Chapter 8). The moral of this story, again, is that simple-looking results may be far from trivial. At a deeper but speculative level, the existence of the inelegant result was evidence that the model is polynomial, which may have

[†] More information on binary max heaps can be found in any standard text on data structures, and in Wikipedia. We are grateful to Han Hoogeveen for pointing out the role of the binary max heap structure.

motivated Lawler to find a better polynomial solution. On a related note, according to Agnetis et al. (2004), the algorithm proposed by Lawler in 1973 was independently published in the Soviet Union by Livshits in 1969, for the basic case without precedence constraints.

The simplicity and usefulness of the MDD result was first established by Baker and Bertrand (1982). An earlier heuristic by Wilkerson and Irwin (1971) yields a sequence that satisfies MDD for all adjacent pairs. Essentially, it involved the same adjacent pair interchange analysis. They also proved a sufficient optimality condition. For each job define the *tardiness interval* as follows:

$$I_j \text{ is empty if } C_j \leq d_j$$

$$I_j = [d_j, C_j], \text{ otherwise}$$

Then in a schedule that is stable under MDD, if the tardiness intervals are nonoverlapping, the solution is optimal. That is, optimality is achieved if there does not exist a time t at which two or more uncompleted jobs are already tardy. More fundamental results on tardiness were published in 1969 by Emmons, as we discuss in Chapter 3. However, the roots of tardiness analysis go back to Smith (1956), although he did not provide sufficient details. Perhaps for this reason, his contribution to this problem was neglected.

Smith's tardiness result demonstrated that no simple sorting rule similar to SPT or EDD can solve the T-problem (as we have also demonstrated in the chapter). This is a "negative" result, whereas the optimality of SPT and EDD for the respective cases is "positive." In practice, negative results are very important (to prevent mistakes). But for publication purposes, positive results usually receive more attention and appreciation. That may be associated with the fact that to prove a negative result all we need in many cases is a counterexample. Because we believe that negative results can be important, we often present them explicitly. Sometimes we do so in the form of propositions. Most of these propositions are well known by professionals but, for the reasons discussed before, not necessarily previously published.

The Standard Notation Scheme

In the scheduling literature, a standard notation introduced by Graham et al. (1979) is often used as a shorthand description of models. Although we do not use this notation in our coverage, it is important for readers who want to be able to read the literature directly.

The notation has three fields, with the general format $\alpha \mid \beta \mid \gamma$, where α indicates the machine environment, β lists job characteristics and γ summarizes the objective function. In the first few chapters we deal with single-machine environments, so we have $\alpha = 1$. The paper by Johnson that we mentioned in the research notes of Chapter 1 concerned a two-machine flow shop. For that model we would use the standard notation $F2$ for the machine environment; Fm would denote a similar flow shop with m machines. A job shop with m machines is denoted by Jm. Similarly, models with m parallel identical machines are denoted by Pm. The β field is often left blank, but it might include notations such as *prec* to denote precedence constraints between jobs or r_j to denote non-zero release dates. Multiple entries may also be necessary in this field. The γ field is typically straightforward, e.g., $\sum w_jF_j$ for weighted flow time or $\sum T_j$ for total tardiness. In this case, multiple entries are rare but could be associated with models that have multiple criteria. We illustrate by giving the notations for some examples that we discussed:

(*i*) Moore (1968) solved the $1 \mid \mid \sum U_j$ problem;

(*ii*) as reported by Moore (1968), Lawler transformed $1 \mid \mid g_{max}$, where g_{max} is the maximal penalty of any job in a sequence (see Chapter 3), to a $1 \mid \mid \sum U_j$ problem (and thus showed that the g_{max} problem is polynomial);

(*iii*) several years later, Lawler (1973) presented a much more elegant solution for $1 \mid prec \mid g_{max}$. For comparison, $1 \mid \mid g_{max}$ is a special case where the set of precedence relationships, *prec*, is empty);

(*iv*) as related by Agnetis et al (2004), the same elegant algorithm for $1 \mid \mid g_{max}$ was presented independently by Livshits (1969).

(*v*) Smith (1956) presented several models, including $1 \mid \mid \sum w_j F_j$ (total weighted flowtime), solved by SWPT, and $1 \mid \mid T_{max}$, solved by EDD;

(*vi*) Jackson (1955) addressed the (NP-hard) maximal tardiness problem with release dates, $1 \mid r_j \mid T_{max}$. (In Chapter 8 we show that this problem is equivalent to $1 \mid r_j \mid C_{max}$.)

(*vii*) Johnson (1954) solved the two-machine flow shop with a makespan objective so the notation would be $F2 \mid \mid C_{max}$. He also discussed some three machine instances, namely, special cases of $F3 \mid \mid C_{max}$.

To illustrate a double entry in the β field, take the model $1 \mid prec, r_j \mid g_{max}$ (which is the case addressed by Lawler, but with release dates). We can show by restriction that this model is NP-hard: $1 \mid r_j \mid T_{max}$, which is NP-hard (see [*vi*]), is a special case.

References

Agnetis, A., P.B. Mirchandani, D. Pacciarelli and A. Pacifici (2004), "Scheduling Problems with Two Competing Agents," *Operations Research* 52(2), 229-242.

Baker, K.R. (1974) *Introduction to Sequencing and Scheduling*, Wiley, New York.

Baker, K.R. (2005) *Elements of Sequencing and Scheduling*, Tuck School of Business, Hanover, NH.

Baker, K.R. and J.W.M. Bertrand, (1981) "An Investigation of Due Date Assignment Rules with Constrained Tightness," *Journal of Operations Management* 1, 109-120.

Baker, K.R. and J.W.M. Bertrand (1981) "A Comparison of Due Date Selection Rules," *AIIE Transactions* 13, 123-131.

Baker, K.R. and J.W.M. Bertrand (1982) "A Dynamic Priority Rule for Scheduling Against Due-Dates," *Journal of Operations Management* 3, 37-42.

Baker, K.R. and J.J. Kanet (1983) "Job Shop Scheduling with Modified Due Dates," *Journal of Operations Management* 4, 11-22.

Conway, R.W., W.L. Maxwell and L.W. Miller (1967) *Theory of Scheduling*, Addison-Wesley, Reading, MA.

Emmons, H. (1969) "One-Machine Sequencing to Minimize Certain Functions of Job Tardiness," *Operations Research* 17, 701-715.

Graham, R.L., E.L. Lawler, J.K. Lenstra and A.H.G. Rinnooy Kan (1979) "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," *Annals of Discrete Mathematics* 5, 287-326.

Jackson, J.R. (1955) "Scheduling a Production Line to Minimize Maximum Tardiness," Research Report 43, Management Science Research Project, University of California, Los Angeles.

Lawler, E.L. (1973) "Optimal Sequencing of a Single Machine Subject to Precedence Constraints," *Management Science* 19, 544-546.

Livshits, E. M. (1969) "Minimizing the maximal penalty in a single machine problem," *Transactions 1st Winter School Mathematical Programming*. Drogobych, Ukraine, 454–475 (in Russian).

Moore, J.M. (1968) "An *n* Job, One Machine Sequencing Algorithm for Minimizing the Number of Late Jobs," *Management Science* 15, 102-109.

Smith, W.E. (1956) "Various Optimizers for Single Stage Production," *Naval Research Logistics Quarterly* 3, 59-66.

Wilkerson, L.J. and J.D. Irwin (1971) “An improved algorithm for scheduling independent tasks,” *AIIE Transactions,* 3(3), 239-245.