Research Notes for Chapter 12[*]

Below we discuss early research in lot streaming with a makespan objective (on which our chapter focuses). We then cover the historical role of the JIT movement in this research. That role is associated with MRP—which had been developed in America at the time JIT was developed in Japan. The advent of JIT revealed that MRP required major enhancements, of which the need for lot streaming models was one. We then show that the problems we identified in the chapter as polynomial are indeed in P, provided we need a discrete solution. This is important because there is a view in the literature according to which the problem is not even in NP (because for arbitrary s it is impossible to write down the solution in polynomial complexity). Next, we turn our attention to models with other objectives and point out related sources. Finally, we generalize the results we reported in Section 12.2.4 for two machines with equal processing times and attached sublot setups to unequal processing times. A comprehensive treatment of lot streaming models can be found in the monograph by Sarin and Jaiprakash (2007).

Sources and Comments

Whereas the term *lot streaming* was coined in the sixties (Reiter, 1966), with few exceptions, explicit attention to lot streaming models started with the advent of Just-In-Time (JIT) in the US in the 1980s (e.g., see the title of Kulonda, 1984). The earliest exception that we should mention is that if we ignore the fact that the optimal solution for several jobs may call for mixing sublots (see Example 12.5, which is due to Potts and Baker, 1989), then in our coverage of Mitten (1959) in Chapter 11 we implicitly considered sublots by modeling overlapping (in a multi-job context). The main distinction is that there was no attempt to study the effect of lot size or limit the number of transfers. Allowing any number of transfers would be appropriate when machines are close by. When transfers are not easy we can take the approach of the chapter—originally espoused by Szendrovits (1975, 1976) in a study of the effect of equal sublots without idling on inventory holding costs—and limit the number of sublots explicitly. Alternatively, we can estimate the cost of each transfer and account for it in the model, so the number of transfers is part of the solution (Goyal, 1976). For instance, we can impose a budget constraint on the total cost of transfers. For two machines, the budget can be translated to a number of sublots, s, but for several machines the results are different. For instance, consider the following three-machine example from Trietsch (1987a, 1989): $U = 5$, $p_1 = 1$, $p_2 = 1$, $p_3 = 5$, $c_{12} = 1$, $c_{23} = 10$, and $B = 23$ (where c_{ij} denotes the transfer cost between machines i and j, and B is the

[*] 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 12 in *Principles of Sequencing and Scheduling* (Wiley, 2019). URL: http://faculty.tuck.dartmouth.edu/principles-sequencing-scheduling/.

budget constraint). Figure RN12.1 depicts the optimal continuous solution, which happens to be integer. The makespan is 27 and the solution requires three transfers between machines 1 and 2 but only two between machines 2 and 3. Intermittent idling is also necessary to achieve the minimal makespan.

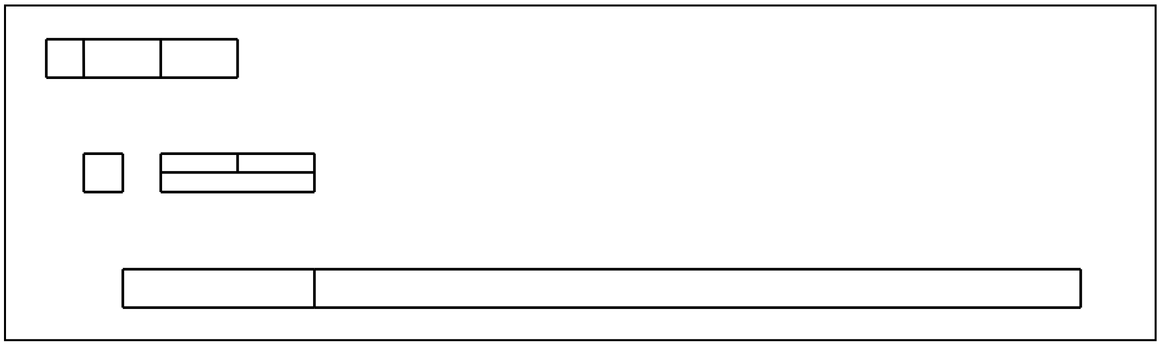

FIGURE RN12.1 Lot Streaming with a Budget Constraint

None of the methods addressed in the chapter can solve this problem. For instance, if we were to use two sublots throughout, the makespan would increase from 27 to 27.5. (The optimal integer makespan is 29.) We may conclude that the budget version of the problem is more complex. Using a budget constraint on transfers does not directly ensure that the plan will be feasible with a given fleet of transporters, but it is conceptually possible to adjust the budget until a feasible plan is achieved. Furthermore, this approach provides a basis for making optimal transportation capacity decisions.

The general problem of transporter scheduling (with or without a capacity sizing element) is an open research question. But basic models for minimizing the makespan of the two-machine case with a given number of transporters, which schedules each transfer, have been published. Truscott (1986) solves the single-transporter case by branch and bound. Trietsch and Baker (1993) solve for any number of transporters efficiently, by extending the discrete lot sizing model covered in the chapter. They show that when the transporter can accommodate the whole batch if necessary, the solution is similar to the basic case, with geometric sublots. The same observation still holds if transporters have limited capacity, but there are enough transporters to support the machines. In that case, some lots can be geometric but other lots have a fixed size as dictated by the transporter capacity. However, if transporters have limited capacity, once their number drops below a particular threshold, makespan *and* capacity (throughput) are dictated by the transporters' capacity. If items have to be processed by an additional resource that takes fixed time between machine 1 and machine 2, such as an oven, the same model applies because such resources are mathematically equivalent to transporters: they take constant time and can serve more than one item concurrently. (In Chapter 13 we discuss the scheduling of such resources in for multiple jobs, but without lot streaming.)

In the chapter, we have relied on other results originally reported in Trietsch and Baker (1993). There, in turn, we reviewed models published mostly in the 1980s and included some material from theretofore unpublished papers, namely Baker (1987), Trietsch (1987b), and Trietsch (1989). Specifically, the fundamental partition was originally defined in Trietsch (1989) to solve the variable sublot *m*-machine model. He also defined an additional partition that allows additional machines relative to the fundamental partition without necessarily increasing the makespan (for the budget version of the problem). In the case discussed in the chapter, however, where n is a given parameter, the solution of the continuous m-machines variable sublot case is obtained by using the fundamental partition. In each part, sublots are geometric with q as given in Theorem 12.2 (even though the context is different). It is also straightforward to generalize the procedure for obtaining discrete lot sizes for each part, similarly to our analysis for three machines in the chapter.

The relevance of the fundamental partition to the consistent sublot case emerged independently. As noted in the chapter, the linear programming approach generalizes to any number of machines. For the special case of three machines, however, a more streamlined solution algorithm was developed by Glass, Gupta and Potts (1994). Later it was extended to m machines by Glass and Potts (1998), to whom Theorem 12.2 is due. Clearly that result relies on the fundamental partition. Example 12.5 is based on Potts and Baker (1989). The heuristic of Section 12.5.2, which also relies on the same partition, is from Baker and Pyke (1990). The experimental results we reported in Section 12.4 are due to Baker and Jia (1993). Little progress has been made analyzing lot streaming in stochastic models, but some encouraging results have been encountered in the simulation results of Jacobs and Bragg (1988) and Smunt, Buss and Kropp (1996). The latter studied the efficacy of transfer lots within a job shop environment, where it is possible that sublots from various jobs will compete with each other. Nonetheless, the use of smaller sublots proved beneficial both for makespan and flowtime measurements. (We return to the flowtime version of the problem later.) In that paper the authors also note that when an attached batch setup is involved, it is useful to send a small initial sublot to enable the setup earlier. Our analysis on page 280, where we showed that the first sublot should be smaller, is compatible with that observation. One important observation in those stochastic simulations is that they apply the lot streaming model, initially developed for one job in a flow shop to a much more general job shop environment with multiple jobs and different routings. Nonetheless, they each show that lot streaming is highly advantageous. Dauzère-Pérès and Lasserre (1997) reached the same conclusion for a job shop with deterministic processing times. They highlighted two important results of their numerical experimentation: (*i*) the makespan can be reduced considerably by very few sublots; (*ii*) using small sublots, it is possible to approach the lower bound that can be easily obtained for job shops by considering the total load on the machine without too much attention to forced idling on bottleneck machines. The latter result essentially states that it is easier to load critical machines fully when the effective size of production batches is reduced. For this purpose, however, it is necessary to allow mixing sublots of different jobs (which implies that we effectively work with a larger number of smaller jobs) and the sequencing heuristic should have a certain level of sophistication. In particular, the authors use a shifting bottleneck procedure (see Chapter 14) to initially sequence jobs, and then they use a second heuristic to create sublots and allow re-sequencing to obtain lot mixing where necessary.

To our knowledge, there has been no research on safe scheduling models with lot streaming, although Trietsch (1987b) comments briefly that rounding up the size of the first lot decreases the probability of intermittent idling on the second machine. In other words, one way to formulate safe scheduling lot streaming models would be to size the first sublot in a manner that balances the need for smooth operations and for a short makespan.

The Role of JIT as Compared to MRP

To return to the 1980s, at that period American manufacturing was in decline whereas the Japanese were ascending. The Japanese strategy was predicated on perfecting repetitive manufacturing (mass production). The American strategy was (and at the time of this writing still is) more focused on non-repetitive manufacturing, with jobs often involving large batches of items. We should clarify, however, that even quintessential mass production industries such as the automobile industry used the batch production model to fabricate parts, and there was no clear incentive to use smaller batches. In repetitive manufacturing, the main driver of batch size is setup time. Suppose, for instance, that a press has to produce five part numbers (products) at an average rate of 120 per eight hour shift. Suppose the production rate (capacity) is 100 items per hour. A quick calculation shows that the press should operate six hours per day to satisfy the requirements of all five part numbers. This requires cyclic production with five setups per cycle. Now suppose that an average setup takes four hours. Without accounting for random problems, each shift only has two free hours for setups. A full cycle has five setups, and thus requires 20 hours of setup. With two free hours per shift, we must make each cycle large enough to supply ten shifts. That calls for average batches of 1200 items. With safety time of just one hour per shift, these batches must be doubled. In practice, such numbers were often exceeded, and each batch might be sufficient to feed the assembly line for several weeks or even months. Cost accounting showed, however, that the inventory holding cost of such batches contributed only a small fraction of the total operation cost, and therefore no problem was perceived.

Under JIT, by contrast, such large batches were considered wasteful. Pioneered at Toyota under the leadership of Taiichi Ohno, a concerted drive was undertaken to reduce batch size, and because the most visible symptom of batch size is WIP, that drive targeted WIP as the manifestation of waste and sought to minimize it. Batch sizes could only be reduced by reducing the total setup time per cycle. That, in turn, required either buying more presses—which was not an economically viable option—or cutting setup times. We already mentioned the technical approach developed for that purpose—SMED—in our Research Notes of Chapter 8. SMED was developed by Shigeo Shingo at the behest of Ohno. It is based on sound industrial engineering concepts implemented by teams of workers. Emphatically, SMED implementation rarely involves automation or hi-tech solutions. However, it takes time to learn how to achieve significant results and a large factory involves many such setups which, together, take a long time to address.

In addition to setup compression, JIT involved the use of *kanbans* to control the total WIP in the system. Kanbans are essentially cards that must be attached to every batch (often they are attached to containers of the right size to hold a batch). In more modern systems a kanban may be an electronic signal, but it is still conceptually attached to a batch. Without a kanban, there is no authorization to start a new batch. A batch that is consumed (typically by a downstream work center) releases its kanban so it can be attached to a new batch, thus authorizing new production. By restricting the number of kanbans, JIT controls the total WIP in the system, and thus it is possible to adjust the WIP level to achieve the best results that balance flowtime and throughput. Importantly, the work centers—also known as *cells*—are typically organized as small flow shops with a U shape. The U shape has two important advantages. First, the areas for input and output to and from the cell, at the ends of the U, are close to each other. As we shall see, this is helpful for dispatching decisions. Second, workers can easily see the whole line and move freely between machines. Kanbans control the flow of materials between cells but within a cell processing is overlapped. Essentially, these systems are job shops with cells acting as machines, so the job shop comprises a collection of small flow shops. Kanbans are used to control the total WIP in the job shop. Incoming batches, with kanbans attached, wait in cell input areas, whereas free kanbans (for the next stage) are posted in the output area.[*] When they wish to start processing a new batch, cell operators dispatch a free kanban for some part number for which they have the necessary inputs. That is, sequencing various part numbers is done by nondelay dispatching, following simple rules based on the urgency of the part number downstream: part numbers with many free outgoing kanbans typically receive priority. Thus, the decision is based on what the operator observes in the output area, but it is constrained by availability of incoming kanbans. Formally, to be eligible for processing, a job must to be present (attached to an incoming kanban) *and* authorized (by a free outgoing kanban). Once the batch selected is inducted into the cell, the now-free input kanban of that part number is sent back to its source. Because cell operators are not authorized to process a batch unless there is a free outgoing kanban, the system may be blocked. Because they simply cannot process a batch if its incoming queue is empty, starvation is also possible. For instance, if a particular cell is a consistent bottleneck we will typically observe a large incoming queue ahead of it, accompanied by a large number of outgoing free kanbans: that is, there is work authorized and available, but the cell cannot handle it fast enough. By contrast, other cells may idle due to lack of supply (e.g., when they follow a bottleneck, directly or indirectly) or lack of demand (e.g., when they feed a bottleneck, directly or indirectly). Finally, we should note that it is especially easy to pursue quality improvements in such a system. The small batches imply that defects, if any, are detected early, while lessons can still be learned and before damage is done to a much larger number of items. As a rule, the continuous pursuit of quality improvements is an integrated part of a full-fledged JIT system.

Although modern JIT plants are computerized, when JIT was developed in the 1960s it did not rely on computers. By contrast, at the same period, to operate well in the large batch environment, major American industrial organizations placed heavy bets on computerized production planning systems known as MRP (Materials Requirements Planning). MRP, and its direct offspring MRP II (Manufacturing Resource Planning), is the forerunner of what is known today as ERP (Enterprise Resource Planning). The purpose

[*] When the output area of a cell is far from the input area of the next cell for a particular kanban type, transporters may be used for the necessary transfers. If containers are involved, the transporter should take full containers from the output area of the upstream cell to the input area of the downstream cell and bring the empty ones back. (In such case, the kanban can be permanently attached to the container; effectively, then, the container is also the kanban.) In such systems, a special transportation kanban is used to signal that transportation is required. Conceptually, however, transportation is just a type of processing so transportation kanbans are just a type of processing kanbans. Note, however, that we would require one such transportation kanban for full containers going downstream and one for empty containers going upstream. Then again, if the production kanban is attached to the container, it is clear that a full container upstream should go downstream and an empty one upstream should travel back.

of MRP is to manage inventories by releasing production orders to the shop floor and purchasing orders to vendors. That has to be done in a coordinated way such that all the parts necessary for an assembled product will arrive by the time they are needed, without holding excessive safety stocks of each and every part number. (Under JIT, these desired outcomes are assured by the kanban system.) There is no question that non-repetitive production on a large scale requires computerized support. Unfortunately, in objective terms (but perhaps justifiably given the state of the art when MRP was developed), the logic behind typical MRP systems is flawed in several important ways (Baker, 1993). Typically, MRP scheduling starts with a due date and schedules release dates (i.e., scheduling backwards). The main problem is an implicit assumption that if we use CON logic (that is, transform the basic CON equation, $d_j = r_j + \gamma$, to an equivalent form, $r_j = d_j - \gamma$), capacity will be sufficient. This is called the *infinite capacity assumption*. Thus, an initial MRP schedule would not involve any attempt to actually sequence jobs. To achieve smooth production planners had to go through a semi-manual iterative step of capacity planning involving flowtime adjustments (by increasing γ) and crude heuristics to shift production from overloaded periods earlier. Too often, this process did not work very well in practice. Another problem was that in terms of timing, scheduling was done in relatively large time buckets. One additional flaw in the logic—the one relevant to our current context—was that overlapping is simply not accounted for in basic MRP. When combined with the use of large time buckets, the scheduled processing time of a batch would span at least as many periods as there were work centers (machines) through which the batch had to be processed. Overlapping was often practiced, but by expediting rather than by following the MRP schedule. The practice of expediting has its own weaknesses, however, which are legion. The typical end result was very high levels of work in progress inventories with a very large proportion of tardy jobs. On top of that, the quality of American products at that time, while perhaps reasonable by previous standards, was no longer competitive with that of Japanese products.

After more than a decade during which Japanese industry perfected JIT while the American competition remained unaware of its existence, advanced Japanese companies achieved substantial results and opened a gap that would take years to close. The competitive edge JIT provided is outside our scope. Suffice it to say that it proved to be quite decisive. One result was that American business, and business schools, started to pay more attention to production and operations management. The Japanese success was—correctly—attributed to the pursuit of total quality and JIT. JIT, however, was simplistically presented as *Zero Inventory* production; that is, the role of SMED was glossed over.[*] Furthermore, SMED is easier to implement in repetitive nature of production—within which it was initially developed. American manufacturers could not always imitate these conditions. Even to the extent SMED applied to their operations, it takes a long time to implement it extensively. Against that backdrop, one response that received a lot of attention was an advanced MRP program called OPT—an acronym for *Optimized Production Technology*—that employed forward scheduling and focused on the

[*] Taiichi Ohno later stated: "To be sure, if we completely eliminate inventories, we will have shortages of goods and other problems. *In fact, reducing inventories to zero is nonsense*. The goal of the Toyota production system is to level the flows of production and goods. In every plant and retail outlet, we strive to have the needed goods arrive in the quantities at the needed time. *In no way is the Toyota production system a zero-inventory system.*" (Ohno and Mito, 1988, p. 25, emphasis added).

busiest machine, or the bottleneck machine. OPT also allowed transfer lots, which encouraged overlapping and thus reduced work in progress relative to traditional MRP. The software was associated with a set of nine optimization rules (Ronen and Starr, 1990), but these rules were initially kept secret and presented as "nine proprietary equations that guarantee optimality." Once published, it became clear that most of the rules were practically identical to JIT principles. There was one notable exception, namely an excessive focus on the bottleneck machine.* Nevertheless, the combination of secret algorithms and secret principles, alongside the false suggestion of optimality, was successful in raising the interest of the research community in the subject.† Specifically, it became clear that MRP should be enhanced (Vollman, 1986; Graves and Kostreva, 1986). The purpose of typical lot streaming papers was to support that development.

On the Computational Complexity of Discrete Lot Streaming Models

Consider the computational complexity of the problem, for a given s. As noted in the literature, just writing the solution down requires complexity of $O(s)$, and formally speaking that complexity is not bounded by a polynomial of the input size. Notice that the input only requires $m + 2$ numbers: U, s, and p_j ($j = 1, \ldots, m$). This implies that the problem is not even in NP, because we cannot check, or even write, the solution in polynomial time. However, that conclusion is misleading. Without loss of generality we could add a constraint that each transfer lot should include at least one item even in the continuous case (Trietsch, 1987b), and it is certainly true in the discrete case. Starting with the two machine case, if we also have $q \neq 1$, then Trietsch (1987b) showed that the number of usable sublots is bounded by $O(\log U)$. Therefore, writing down the solution takes $O([\log U]^2)$—where the square reflects the fact that it may take up to $O(\log U)$ to write down a lot size. But it also takes $O(\log U)$ to write down the number U itself, and that number is part of the minimal input needed to state the problem. Therefore, for $q \neq 1$, the practical complexity of writing down the solution is polynomial in input size after all. Because we can find the optimal integer solution in time polynomial in s, the discrete version is in P. Now assume $q = 1$, and in the continuous case all sublots are equal. Therefore, again, the solution can be written down sufficiently compactly and the problem is in P. Moving on to the discrete case with $q = 1$, even if U does not divide by s, we can obtain an optimal integer solution by rounding the first few sublots up and the others down. Calculating the number of lots to

* In Chapter 14 we discuss the Shifting Bottleneck Algorithm, which is a highly effective optimization approach for deterministic job shops. Essentially, this algorithm recognizes that it is effective to focus on the bottleneck machine but it is essential to recognize that the identity of the bottleneck machine shifts as we proceed through the algorithm. That algorithm demonstrates that the identity of the bottleneck can shift even during a procedure designed for deterministic processing times. In stochastic environments bottlenecks are even more likely to shift. When we say that OPT placed an excessive focus on the bottleneck, we essentially mean that it failed to recognize that the bottleneck can shift. See Trietsch (2005a; 2005b; 2007) for a detailed discussion of the relevant pitfalls and remedies.

† Eventually, one disappointed industrial customer, MARS, won a law suit against the OPT vendors (Creative Output, Inc.), forcing them to reveal the details of their algorithms to MARS, so MARS could prove that the marketing claims that induced them to purchase the software were false (Wilkins, 1984). Perhaps as a result, these false claims subsequently stopped.

be rounded up takes constant time and writing the solution down remains compact.[*] In other words, a solution that is polynomial in s is in P for any q.

Trietsch (1989), addressing variable sublots, showed the same result, that the number of usable sublots is $O(\log U)$, for each part of the fundamental partition of the m-machine case. So for any given m we again obtain an output size polynomial in input size, and the problem is again in NP. Now consider the consistent sublot version with m machines. Recall from Theorem 12.2 that optimal sublots for the continuous case follow the same geometric ratio structure on successive parts of the fundamental partition. Therefore, the proof that the problem is in NP can be extended. Specifically, each part of the fundamental partition by itself would dictate either $O(\log U)$ transfer lots or that part would contribute a number of equal sublots that can be written down compactly (as x sublots of y each). So even if each part would by itself contribute the maximal s associated with it, we could still write the full solution in time polynomial in input size. In this connection, we reported that the benefit from the first sublot (increasing s from 1 to 2) accounts for more than half of the full potential. This result has been shown independently by Trietsch (1987b, 1989) for variable sublots and by Potts and Baker (1989) for consistent sublots. Thus both models require a similar number of sublots and in both, adding a transfer has a similarly decreasing marginal effect.

Lot Streaming with Flowtime Minimization Objective

The total flowtime objective is usually formulated as a nonlinear measure, rendering its optimization difficult. Chang and Chiu (2005) provide an extensive survey of lot streaming models, including ones with flowtime objectives. Some flowtime results can also be found in Sen et al. (1998). Notably, for flowtime reduction, equal sublots with intermittent idling are quite effective (Kropp and Smunt 1990). They also have an advantage of simplicity. To see that, consider the effect of geometric sublots on flow time. If the second machine is faster, the first sublot is the largest and reaches it late, whereas with equal sublots the first lot arrives earlier. Because we allow intermittent idling on the second machine, it will also complete the first lot earlier. If the second machine is slower, the advantage of using equal sublots is less pronounced; the use of equal sublots will increase flowtime in early sublots but improve it downstream, for an overall approximately balanced effect. In the latter case, intermittent idling does not help (unless the first machine could be utilized for another job during its idling periods). Bukchin, Tzur and Jaffe (2002) address the flowtime version of the two-machine problem with lots setups and teardowns, which imply batch availability. Although there is no simple solution of the type we developed for the makespan problem, the problem can be solved by generic IP. Bukchin and Masin (2004) consider the makespan

[*] In this connection we should mention that at least for large lots there is another option: to round the lot size and change the batch size to s times the rounded lot size. Whereas this solution is outside the box defined by the formal model, in practice one cannot guarantee in advance that starting a batch of, say, 10000 items will yield precisely 10000 items of acceptable quality. To respond to that fact of life, a typical commercial contract allows small deviations in the precise number delivered. In practice, during the era in which basic sublot models were developed, deviations of 5% in batch size were often acceptable. Therefore, using explicit or implicit long and short penalties, the critical fractile model can be used to decide the optimal actual batch size required for servicing the nominal batch size. In such a system, if we have to produce 10100 items instead of 10000, then if we decide to use nine lots of 1122 each, the fact that 10100 does not divide by nine is insignificant: even a lot size of 1125 is not likely to lead to appreciably different expected economic cost.

and the flowtime objectives together. A tradeoff exists between the two, so Pareto-optimal solutions may be identified. Again, a very effective heuristic for the purpose of achieving a good makespan without increasing the flowtime too much is the use of equal sublots with intermittent idling.

Lot Streaming with Total Cost Objective

Nearly all of the literature on lot streaming focuses on traditional scheduling objectives, such as makespan or flowtime. An exception is due to Langevin et al. (1999). They formulate an objective function that contains four components, each representing a source of direct cost in a typical flexible manufacturing system. The four components are: material handling cost, pallet costs, inventory holding cost, and machine cost. The first two components are related to the number of round trips by the material handler, and the other components are related to time that work in progress spends in the system.

The Case of Attached Sublot Setups with Two Unequal Machines

Early models with setups appeared in Truscott (1985), Trietsch (1987a) and in Baker (1988). Here we show how our generic approach can solve the two-machine makespan model with attached sublot setups and teardowns of the type considered by Bukchin, Tzur and Jaffe (2002). This analysis expands our coverage of Subsection 12.2.4 where we assumed the two machines have the same production rate ($q = 1$). We now show the analysis for the $q \neq 1$ case with attached setups (and thus with sublot availability). Our approach is similar to the one adopted in the chapter for the regular case. For convenience, let h denote h_2; that is, $h = (SU_2 - SU_1)/p_1$. In addition to the notation given in the chapter, let $q_r = p_1/p_2$ (= $1/q$)—i.e., it is the value of q as defined for the reversed problem—and, similarly, let $h_r = (SU_1 - SU_2)/p_2$. (By this definition, if $h > 0$ then $h_r < 0$ and vice versa.) By arguments similar to those we used in the chapter for the no-setup case, all sublots in an optimal solution must be critical, and thus there should be no waiting of sublots completed on machine 1 for processing on machine 2, and there should be no idleness on machine 2 between sublots. Therefore, L_j should start its setup on machine 2 when L_{j+1} starts its setup on machine 1 (for $j = 1, 2, \ldots, n - 1$); and, similarly, L_j completes its teardown on machine 2 when L_{j+1} completes its teardown on machine 1. For our case, $q \neq 1$, this leads to the following ($n - 1$) equations:

$$L_{j+1} = h + qL_j = h\frac{1-q^j}{1-q} + L_1 q^j \quad ; \quad q \neq 1, j = 1,2,\ldots,n-1 \qquad (1)$$

Recall from the chapter that for $q = 1$, optimal sublots form an arithmetic series. Equation (1) indicates that when processing times are different, sublots form a sum of an arithmetic series and a geometric series, and thus they reflect a more general case than any we encountered so far. However, Equation (1) is not sufficient to solve the problem: we seek n unknowns L_j but with only ($n - 1$) equations. The additional equation we need is given by $\Sigma L_j = U$, yielding,

$$L_1 = \frac{1-q}{1-q^n}\left(U - \frac{nh}{1-q} + \frac{(1-q^n)h}{(1-q)^2} \right) \quad ; \quad q \neq 1 \tag{2}$$

We can find L_j for any j by substituting L_1 in Equation (1), recursively. Thus, for any given n and U, along with SU_1, SU_2, p_1, and p_2, the equations above determine the series $\{L_j\}_{j=1,\ldots,n}$ uniquely. Accordingly, we may address L_j as a function of n and U, $L_j(n, U)$; but we suppress the argument U if it is given (which is usually the case). We need to allow n to vary because our task remains to find the optimal n to minimize the makespan. When n is too large (or U too small) one or more of the constraints $L_j \geq 0$, which we ignored so far, may be violated. However, that cannot happen if both L_1 and L_n are nonnegative. To check both without solving all intermediate values, we can find L_n by symmetry; that is, to find L_n we can use q_r and h_r instead of q and h in Equation (2). It can be shown, however, the optimal solution obtained without considering this constraint is guaranteed to be feasible, so we can safely ignore these constraints. Formally, we will say that n is *admissible* if the n equations associated with it yield $L_j \geq 0$ for both L_1 and L_n, (and thus for all j).

Because there is no need for intermittent idling, the makespan is given by the time to complete L_1 on machine 1 (including one setup) plus the time to process the whole batch on machine 2 (including n setups). The symmetric makespan expression should yield the same value, with L_n taking the place of the first sublot and the machine indices reversed, thus leading to the following result.

$$M = SU_1 + L_1 p_1 + nSU_2 + Up_2 = SU_2 + L_n p_2 + nSU_1 + Up_1 \tag{3}$$

By substituting Equation (2) for L_1, this expression becomes an explicit function of n. Given some admissible n value, all sublots are uniquely determined and thus the makespan is also given. Because U is a known parameter, our problem boils down to the optimal selection of n, which can be done in various ways, including trial and error. A useful approach is to treat U as a variable and study the conditions under which some fixed n is optimal. Clearly, as we increase U, $L_j(n, U)$ increases for all j, and vice versa. One advantage of this approach is that if production of this particular product is repetitive, we can tabulate the results so we won't need to solve repeatedly for every possible U. This intuitive result can be formally verified by observation of (1): for example, it includes elements of the form $L_1 q^{j-1}$, which all increase with L_1, as well as a constant function of A. As long as we keep n constant, any increase in U must be apportioned to the L_j values as proportional increases to all the elements $L_1 q^{j-1}$. Similarly, for a given n, if we add a transfer the former sublots must decrease, and if we cancel a transfer the remaining sublots must increase, i.e., $L_j(n-1, U) > L_j(n, U) > L_j(n+1, U)$ for all relevant j. To see this formally it may be convenient to select the primal version of the problem if $h \geq 0$, and the reversed if $h_r > 0$. Therefore, we can assume without loss of generality that $h \geq 0$. When we add (delete) a transfer lot, the effect is a reduction (increase) in the processing time of L_1 accompanied by the addition of SU_2 to the total setup and processing time on machine 2. Suppose now that the following two inequalities hold:

$$L_1(n, U) - L_1(n+1, U) \leq \frac{SU_2}{p_1} \tag{4}$$

$$L_1(n-1,U) - L_1(n,U) \geq \frac{SU_2}{p_1} \tag{5}$$

then n must be optimal for U: The first inequality requires that adding a sublot will not decrease L_1 enough to compensate for the addition of SU_2, since it will only reduce the processing time of L_1 on machine 1 by less than SU_2. Thus, n is better than $n+1$ if this inequality holds strictly, and it is indifferent if it holds as an equality. The second inequality, similarly, requires that reducing the number of sublots by 1 will not decrease the makespan because the additional processing time on machine will exceed SU_2. Thus, the two conditions together suffice for n to be optimal. We can find the optimal n by a standard search procedure that takes $O(\log U)$. Notice that equations (4) and (5) can easily be used to determine not only if n is optimal but also, if it is not optimal, whether it is too small or too large. For example, when the left hand side of (4) is larger than the right hand side, n should be increased, and, similarly, when the left hand side of (5) is smaller than the right hand side, n should be decreased. Therefore we can find the optimal n by bisection search.

Numerical example: Let $U = 75$, $p_1 = 2$, $p_2 = 3$, $SU_1 = 6$, $SU_2 = 16$. Using the $q = 1$ result from the chapter as an approximation leads to

$$n = \sqrt{\frac{75(2+3)}{6+16}} = 4.129 \approx 4$$

Equation (4) is satisfied by $n = 4$, but by Equation (5), $n = 4$ is excessive; thus it is clear that $n = 3$ satisfies Equation (4). Checking Equation (5) again we find that $n = 3$ is optimal. The optimal CV solution with 3 sublots leads to a makespan of 303.2105. The solution with $n = 4$ is close, however, at 303.3076. For $n = 2$, we obtain 319 (and this case happens to yield an integer solution with lots of 28 and 47) and $n = 5$ *would* yield 309.9573 (i.e., higher than 303.2105) but it is infeasible in the sense that the first lot has to be negative to achieve this makespan, so the true minimal makespan with $n = 5$ must be higher. In general, it can be shown that the optimal n never leads to a negative sublot.

We can generalize the algorithm from the chapter to find the best integer solution. However, we can only use it determine integer lot sizes for a given n, and we may need to employ trial and error to identify n^* for the discrete case. This is not difficult in practice because the discrete n^* tends to be close to the continuous one. For that reason, it is reasonable to start the algorithm with $n = n^*$ as determined in the continuous case.

```
0.    For a given $n$, let $j = 0$, $S_0 = 0$, with $M$ as obtained for the continuous case.

1.    Set:  $j = j + 1$
            $S_j = \min\{\max\{\lfloor(M - p_2(U - S_{j-1}) - jSU_1 - (n - j + 1)SU_2)/p_1\rfloor, 0\}, U\}$
            $f_j = \min\{\max\{(M - p_2(U - S_{j-1}) - jSU_1 - (n - j + 1)SU_2)/p_1, 0\}, U\} - S_j$
            $e_j = 1 - f_j$
```

2. If $j < n$, return to step 1; else, continue

3. If $S_n = U$, STOP (the current value of M is optimal for n); else continue to step 4

4. Set $j = 0$ and $M = M + p_1 \min_j\{e_j\}$; return to step 1.

Again, we cannot guarantee that the optimal discrete solution utilizes the same number of transfer lots as the continuous one. The optimal numbers of transfers for the continuous version and the discrete version tend to be close to each other when the setups are large. Otherwise, the difference between them can be reduced if we utilize a version that ensures $L_j \geq 1$, without which the theoretical optimal n can even exceed U. On the one hand, n is bounded by $U / \min\{SU_1, SU_2\}$ but, on the other hand, with zero setups we obtain $n = \infty$, leading to a makespan of $U\max\{p_1, p_2\}$. The algorithm, however, requires n as input. In practice, we should check the neighboring n values. But it is not necessary to solve for all these values: using branch and bound logic, it is enough to check whether the optimal M for the current n is feasible for those neighbors for which the continuous minimal makespan does not exceed the current M. That is, once we get one solution for one n value, typically using the optimal continuous version n, it can serve as an upper bound on the optimum and there is no need to test any version whose lower bound obtained by the continuous version is higher.

Numerical example (continued)

Performing the computations for $n = 3$ starting with the trial makespan of 303.2105, we obtain $L_1 = 12$ with a fraction f_1 of 0.10525 and a complement $e_1 = 0.89475$. Similarly, the next two lots are 23 and 39 with e_j values of 0.89475 and 0.39475. $S_3 = 74 < 75$. Therefore, we should add $p_1\min\{e_j\} = 2 \times 0.39475$ to M, yielding a new trial value of 304. This leads to the optimal solution with $L_1 = 12$, $L_2 = 23$, $L_3 = 40$. As it happens, the same optimal solution can also be achieved with $n = 4$ sublots of 4, 11, 22, and 38. However, 304 is less than the continuous makespans for $n = 2$, $n = 4$, and $n = 5$ so it follows that $n = 3$ and $n = 4$ are globally optimal. That $n = 4$ is suboptimal for the continuous case but optimal for the discrete case indicates that there is no guarantee the optimal n values for the two cases coincide. Finally, the solution with $n = 4$ is better in terms of flow time: the flowtime associated with $n = 3$ (without intermittent idling) is 17020, as compared to 16564 for $n = 4$.

References

Baker, K.R. (1987) "Lot Streaming to Reduce Cycle Time in a Flow Shop," Working Paper #203, The Amos Tuck School of Business Administration, Dartmouth College, Hanover, NH.

Baker, K.R. (1993) "Requirements Planning," Chapter 11 in *Logistics of Production and Inventory* (S. Graves, A. Rinnooy Kan and P. Zipkin, eds.), Handbooks in Operations Research and Management Science, Vol. 4, Elsevier.

Baker, K.R. (1995) "Lot Streaming in the Two-Machine Flow Shop with Setup Times," *Annals of Operations Research* 57, 1-11.

Baker, K.R. and D.F. Jia (1993) "A Comparative Study of Lot Streaming Procedures," *Omega* 21, 561-566.

Baker, K.R. and D.F. Pyke (1990) "Solution Procedures for the Lot Streaming Problem," *Decision Sciences* 21, 475-491.

Bukchin, J., M. Masin (2004) "Multi-Objective Lot Splitting for a Single Product *m*-Machine Flowshop Line," *IIE Transactions* 36, 191-202.

Bukchin, J., M. Tzur and M. Jaffe (2002) "Lot Splitting to Minimize Average Flowtime in a Two-Machine Flow Shop," *IIE Transactions* 34, 953-970.

Dauzère-Pérès, S. and J.-B. Lasserre (1997) "Lot Streaming in Job-Shop Scheduling," *Operations Research* 45, 584-595.

Glass, C.A., J. Gupta and C.N. Potts (1994) "Lot Streaming in Three-Stage Production Processes," *European Journal of Operational Research* 75, 378-394.

Goyal, S.K. (1976) "Note on 'Manufacturing Cycle Time Determination for a Multi-stage Economic Production Quantity Model,'" *Management Science* 23, 332-333.

Graves, S.C. and M. Kostreva (1987) "Overlapping Operations in Material Requirements Planning," *Journal of Operations Management* 6, 283-294.

Jacobs, F.R. and D.J. Bragg (1988) "Repetitive Lots: Flow-time Reductions through Sequencing and Dynamic Batch Sizing," *Decision Sciences* 19, 281-294.

Kropp, D.H. and T.L. Smunt (1990) "Optimal and Heuristic Models for Lot Splitting in a Flow Shop," *Decision Sciences* 21, 691-709.

Kulonda, D.J. (1984) "Overlapping Operations—A Step Toward Just-In-Time Production," *Readings in Zero Inventory*, APICS 27th Annual International Conference, 78-80.

Langevin, A., D. Riopel, and K.E. Stecke (1999) "Transfer Batch Sizing in Flexible Manufacturing Systems," *Journal of Manufacturing Systems* 18, 140-151.

Mitten, L.G. (1959) "Sequencing n Jobs on Two Machines with Arbitrary Time Lags," *Management Science* 5, 293-298.

Ohno, T. and S. Mito (1988) *Just-In-Time for Today and Tomorrow*, Productivity Press, Cambridge, MA.

Potts, C.N. and K.R. Baker (1989) "Flow Shop Scheduling with Lot Streaming," *Operations Research Letters* 8, 297-303.

Reiter, S. (1966) "A System for Managing Job Shop Production," *Journal of Business* 34, 371-393.

Sarin, S.C. and P. Jaiprakash (2007) *Flow Shop Lot Streaming*, New York: Springer.

Szendrovits, A.Z. (1975) "Manufacturing Cycle Time Determination for a Multi-Stage Economic Production Quantity Model," *Management Science* 22, 298-308.

Szendrovits, A.Z. (1976) "On the Optimality of Sub-Batch Sizes for a Multi-Stage EPQ Model-A Rejoinder," *Management Science* 23, 334-338.

Trietsch, D. (1987a) "Optimal Transfer Lots for Batch Manufacturing," Presentation at the ORSA/TIMS Conference, St Louis, October.

Trietsch, D. (1987b) "Optimal Transfer Lots for Batch Manufacturing: A Basic Case and Extensions," Technical Report NPS-54-87-010, Naval Postgraduate School, Monterey, CA.

Trietsch, D. (1989) "Polynomial Transfer Lot Sizing Techniques for Batch Processing on Consecutive Machines," Technical Report NPS-54-89-011, Naval Postgraduate School, Monterey, CA.

Trietsch, D. (2005a) "Why a Critical Path by any Other Name would Smell Less Sweet: Towards a Holistic Approach to PERT/CPM," *Project Management Journal* 36, 27-36.

Trietsch, D. (2005b) "From Management by Constraints (MBC) to Management by Criticalities (MBC II)," *Human Systems Management* 24, 105-115.

Trietsch, D. (2007) "System-Wide Management by Criticalities (MBC II): Hierarchical Economic Balancing of Stochastic Resources," *Human Systems Management* 26, 11-21.

Trietsch, D. and K.R. Baker (1993) "Basic Techniques for Lot Streaming," *Operations Research* 41, 1065-1076.

Truscott, W.G. (1985) "Scheduling Production Activities in Multi-stage Batch Manufacturing Systems, *International Journal of Production Research* 23, 315-328.

Truscott, W.G. (1986) "Production Scheduling with Capacity Constrained Transportation Activities," *Journal of Operations Management* 6, 1986, 333-348.

Wilkins, B. (1984) "Judge Orders Software Firm to Hand Over Source Code," *Computerworld* (July 9), 2.