Chapter 19
Advanced Production Planning and Scheduling Systems

Henk Zijm and Marco Schutten

Abstract In this chapter, we present algorithms for a number of functions of the production planning framework presented in Chap. 5. We focus on models for integrated Capacity and Master Production Planning, Job Planning and Resource Group Loading, and Shop Floor Scheduling and Control. At the Master Production Planning level, we exploit a simple Linear Programming formulation to set appropriate capacity levels and in particular to decide whether a temporary expansion of capacity is needed (e.g., through overtime work). With the same formulation, we decide what end-items are to be produced in which period. By applying the lead time offset procedure that is the heart of Materials Requirements Planning, and using the Bill of Materials information, the same is done on the level of part manufacturing (basic level). Essential in the above procedure are two parameters, the effective overall capacity of each manufacturing shop and the final assembly department, often indicated as the maximum throughput, and the lead times needed to complete a part or product in each department. A significant portion of these lead times may in fact be waiting times in front of individual workstations that are busy. To minimize these waiting times, workload control norms are often used which in turn may influence the effective capacity. An essential question then is what these workloads should be in order to match a desired throughput and production lead time. That question is answered by exploiting a Closed Queueing Network approach that explicitly determines the relation between a preset work-in-process level, throughput and the resulting lead times (advanced level). Finally, we exploit a detailed shop floor scheduling procedure, called the Shifting Bottleneck approach, that basically serves to ascertain that internal due-dates, following from the above defined internal manufacturing lead times are indeed met (state-of-the-art).

H. Zijm (✉) · M. Schutten
Department of Industrial Engineering and Business Information Systems, University of Twente, Enschede, Overijssel, The Netherlands
e-mail: w.h.m.zijm@utwente.nl

M. Schutten
e-mail: m.schutten@utwente.nl

H. Zijm et al. (eds.), *Operations, Logistics and Supply Chain Management*,
Lecture Notes in Logistics, https://doi.org/10.1007/978-3-319-92447-2_19

19.1 Introduction: Setting the Stage (*Basic*)

In this section, we consider a generic manufacturing facility, consisting of a parts manufacturing shop followed by an assembly department. Typically, the number of parts produced is limited while a variety of end products can be assembled from specific parts configurations as specified by the Bill of Materials (BoM) of the final product. We assume that purchased items and raw materials needed for parts production can be stocked and that a small inventory of finished parts exists, from which they are picked to enter a final assembly stage. Depending on specific product-market combinations, final products are either produced on customer order or distributed to sales outlets to meet future market demand. Note that, if final assembly is based upon confirmed customer orders, finished product stocks will in principle not exist (apart from a possible small delay in shipping them to the customer); in that case we are dealing with a make-to-stock, assemble-to-order system. A Customer Order Decoupling Point (CODP) is thus located at the finished parts storage facility. If some parts are already customer specific, the intermediate stock point for these parts is also removed, and the CODP is shifted further upstream. However, in order to describe the most generic situation, we in principle accept stock after any possible stage, while in addition Work-in-Process (WIP) inventories are obviously present on the manufacturing shop and assembly floor (cf. Fig. 19.1).

Parts processed in the parts manufacturing shops need a certain lead time from picking the necessary materials or purchased items to be used, until their completion and placement in the appropriate finished part stock point (each part type has its own stock location). Before beginning the assembly of a final product, the parts specified by the Bill of Materials are collected from the intermediate storage and subsequently assembled, which again requires a certain lead time. As we will see later, the length of the lead time strongly depends on the workload of the shop; a shop facing a high workload will generally reveal longer lead times, because parts to be processed on a highly utilized machine may experience longer wait times. The impact of lead times on the overall workload is discussed in Sect. 19.4.

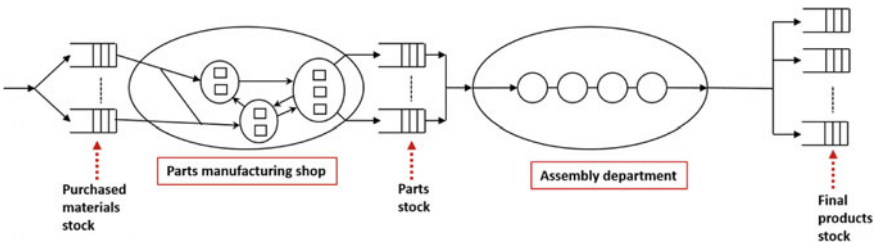

Fig. 19.1 Two stage manufacturing facility

19.2 Case Study: Injection Molding Machinery (IMM)

Injection Molding Machinery (IMM) in the Netherlands is part of one of Europe's larger manufacturers of capital goods and industrial equipment. IMM was founded in 1968 and quickly specialized in the manufacturing of innovative injection molding machines. A typical machine costs between 0.5 million and 2 million Euro. The world market share is about 4%. Some 80% of all machines are exported to Germany. Most of the about 1300 components required in the assembly of the molding machines are purchased. Only 50 components—the more voluminous part types—are processed in the firm's own manufacturing department. All purchasing and manufacturing activities are customer order driven.

When the production and selling of a new range of injection molding machines began, and a significant increase in demand was expected, the firm decided to install a Flexible Manufacturing Cell (FMC) in their manufacturing department as a replacement for two horizontal milling machines. The cell consisted of a machining center with a 110-slot capacity tool magazine, a parts pallet storage with a capacity of 6 FMC pallets, a rail-guided pallet transport vehicle, and a clamping/unclamping station. Both the exchange of cutting tools in the tool magazine and the clamping and unclamping of parts on the pallets were done manually. A hoist is installed for the transportation of the voluminous and heavy parts to and from the clamping/unclamping station.

The parts assigned to the FMC often need some customer-specific processing. Therefore, the number of different NC programs increases steadily. To date, there are over 1000 NC programs for about twenty part type families, which need over 200 different cutting tools. NC programs are written in the Process Planning department.

The Manufacturing Planning and Control system of the firm consists of three hierarchical levels: a Master Planning Level, a Production Management Level and a Shop Floor Control Level. The Master Planning Level is basically responsible for generating orders for the FMC. On a strategic level, it is decided that 2–4 injection molding machines per week can be assembled and shipped to the customer. Each injection molding machine has customer-specific parts (mostly variants of specific part types). Engineering makes drawing for these parts. Process Planning subsequently determines the routings of the parts through the manufacturing department and estimates the required processing times. After this, the BoM of a particular injection molding machine is known, and the information will be processes by a Manufacturing Resource Planning (MRP) system. Purchasing of components and raw materials, parts manufacturing and machine assembly are all customer-order driven. The estimated planned lead times for assembly, manufacturing and purchasing are 5, 2 and 6 weeks respectively. Manufacturing is planned such that all the components

for a particular machine are available and can be kitted two weeks before the assembly of the particular injection molding machines is to occur. These two weeks are the safety lead time and are built in because of problems with the delivery of raw materials.

The second level of the production control system is the Production Management Level. At this level, an important objective is to make sure that the planned lead times are met in an efficient way. Therefore, the production management level must tune the manufacturing capacity into the order flow. The Production Management Level receives weekly information from the MRP system about the capacity needed for each machine, including the FMC, covering a period of eight weeks. Based on this information, capacity decisions are made regarding overtime, weekend work, and/or subcontracting.

The third level of the production control system is the Shop Floor Control Level. Input to this level is a daily MRP list of orders. During the night, the MRP list is automatically updated. Orders are sequenced according to a critical ratio rule that considers internal due dates (that is, the day manufacturing is scheduled to be finished) and the cumulative processing time required for an order in the manufacturing department. Operators must process the orders as much as possible according to the given sequence. The order on the MRP list of the FMC represents a cumulative workload of one, two or sometimes even three weeks of the machining center.

The company faces various problems. There is a pressure to decrease the total lead time, starting with a reduction of the safety time from two to one week. In addition, it lacks an adequate tool to relate the workload to the realized internal manufacturing shop lead times, while at the shop floor level, a more sophisticated sequencing rule is needed. Currently too many deviations of the prescribed sequence occur due to unavailability of raw materials, part programming that is not completed in time, or unavailability of fixtures and cutting tools (that are still in use by other jobs). The company therefore is in need of a sound planning and scheduling mechanism.

19.3 Integrated Capacity and Master Production Planning (*Basic*)

In this section, we present a model for the formulation of an integrated capacity and master production planning problem for a finite planning horizon of T periods (say T weeks). We start with a single stage manufacturing shop and later extend the model to a two-stage system, consisting of a parts manufacturing department followed by an assembly shop. We assume that periodic demand can be reliably forecasted for the full planning horizon. Since capacity is generally limited and in general not sufficient

to meet peak demands in case of a volatile market or e.g. in case of strong seasonality patterns, it may be needed to start production of some products early and to store them temporarily, to make sure that eventually all demand can be met. In such as case, inventories basically serve as shifted (allocated) capacities, as opposed to cases where they serve to anticipate normal demand fluctuations (for which safety stocks are generally held).

We start with some notations. Let t denote the time period index, $t = 1, \ldots, T$. We consider a range of N products, indexed by j, and a manufacturing shop consisting of M workstations, indexed by m. By p_{jm} we denote the processing time, in hours, of one unit of product j on workstation m. The capacity of workstation m in period t is limited by B_{mt} hours. With h_j we denote the inventory costs of one unit of product j per period, while c_{jm} are the cost of processing one unit of product j on machine m. Inventories are calculated at the end of each period. The total lead time of an order of products j through the manufacturing shop equals L_j periods. If an order is placed, it is released at the beginning of a period, say t, and upon completion is added to the stock of product j at the beginning of period $t + L_j$. To describe the routing of jobs through the system, index $a_{jtms} = 1$ if order j released in period t is processed on workstation m in period $s (s \geq t)$, while $a_{jtms} = 0$ otherwise. Finally, let the decision variables Q_{jt} denote the size of an order for products j to be released at the beginning of period t, let I_{jt} be the physical inventory of final products j at the end of period t, and let D_{jt} be the demand for product j which occurs during period t. The following Linear Programming (LP) formulation seeks to minimize the overall production and inventory costs while predicted demand should be met. Since capacity constraints cannot be violated this may cause production occasionally to start relatively early in order to build up inventory such that also peak demands are met.

$$\min \left[\sum_{j=1}^{N} \sum_{t=1}^{T} h_j I_{jt} + \sum_{j=1}^{N} \sum_{t=1}^{T-L_j} \left\{ h_j \frac{1-\alpha^{L_j}}{1-\alpha} Q_{jt} + \sum_{m=1}^{M} c_{jm} Q_{jt} \right\} \right]$$

subject to

$$\begin{aligned}
& I_{jt} = I_{j,t-1} + Q_{j,t-L_j} - D_{jt}, && j = 1, \ldots, N;\ t = L_j + 1, \ldots, T; \\
& I_{jt} = \bar{I}_{jt}, && j = 1, \ldots, N;\ t = 1, \ldots, L_j; \\
& \sum_{j=1}^{N} \sum_{s=t-L_j+1}^{t} Q_{js} p_{jm} a_{jsmt} \leq B_{mt} && m = 1, \ldots, M;\ t = 1, \ldots, T; \\
& I_{jt} \geq s_{jt} && j = 1, \ldots, N;\ t = 1, \ldots, T; \\
& Q_{jt} \geq 0 && j = 1, \ldots, N;\ t = 1, \ldots, T.
\end{aligned}$$

Note that orders for product j released in period t are driven by the demand forecast of period $t + L_j$ period and update inventory only then. With a time horizon of T periods, period $T - L_j$ therefore offers the last opportunity to release an order for product j. Also, since production of a batch Q_{jt} of product j takes a lead time L_j,

it contributes to the work-in-process materials at the manufacturing floor during L_j subsequent periods, for which inventory costs are incurred. We value the batch Q_{jt} at a fraction α^{k-1} $(0 < \alpha < 1)$ of its final value if it still has $k-1$ periods to go till completion $(k = 1, \ldots, L_j)$ and hence update the inventory costs accordingly. The contribution of order Q_{jt} to the work-in-process related inventory costs in the objective function therefore equals

$$h_j\big(\alpha^{L_j-1} + \alpha^{L_j-2} + \cdots + \alpha + 1\big)Q_{jt} = h_j \frac{1-\alpha^{L_j}}{1-\alpha} Q_{jt}$$

which comes next to the periodic inventory costs associated with stored items. The last term in the objective function reflects the production costs. In fact, one can argue that the latter costs are constant since all demand should be produced anyhow and the costs are time independent, but later we will encounter time-dependent production costs and to ease the presentation we therefore add them already here.

If demand is highly volatile or follows a strong seasonality pattern it may be that, despite the possibility to work in advance and store finished products temporarily, the available capacity is still insufficient to meet high demands in some periods. In such a case, companies seek to outsource part of their production or may decide to work in overtime (e.g. a temporary night shift). Obviously, work in overtime is more expensive than work in regular time but on the other hand one may save inventory costs. When allowing for overtime work, one needs to decide which workstation operations (and possibly how much of each operation) are processed in regular time and in overtime. To that end, we define two nonnegative decision variables Q_{jtms} and Q^*_{jtms} that denote the part of order Q_{jt} that is produced on machine m in time period s in regular time and overtime, respectively. Let furthermore B^*_{mt} denote the maximum available overtime capacity of machine m in period t, while c^*_{jm} are the costs of processing one unit of product j at machine m in overtime. Production limitations are then defined by the following two capacity constraints:

$$\sum_{j=1}^{N} \sum_{s=t-L_j+1}^{t} Q_{jsmt} p_{jm} \le B_{mt} \quad m = 1, \ldots, M;\ t = 1, \ldots, T$$

$$\sum_{j=1}^{N} \sum_{s=t-L_j+1}^{t} Q^*_{jsmt} p_{jm} \le B^*_{mt} \quad m = 1, \ldots, M;\ t = 1, \ldots, T$$

while furthermore

$$Q_{jtms} + Q^*_{jtms} = Q_{jt} a_{jtms} \quad j = 1, \ldots, N;\ t = 1, \ldots, T;\ m = 1, \ldots, M;\ s = t, \ldots, t + L_j - 1;$$

$$Q_{jtms}, Q^*_{jtms} \ge 0 \qquad j = 1, \ldots, N;\ t = 1, \ldots, T;\ m = 1, \ldots, M;\ s = t, \ldots, t + L_j - 1.$$

The objective function now becomes

$$\min\left[\sum_{j=1}^{N}\sum_{t=1}^{T} h_j I_{jt} + \sum_{j=1}^{N}\sum_{t=1}^{T-L_j}\left\{h_j \frac{1-\alpha^{L_j}}{1-\alpha} Q_{jt} + \sum_{m=1}^{M}\sum_{s=t}^{t+L_j-1}(c_{jm}Q_{jtms} + c^*_{jm}Q^*_{jtms})\right\}\right]$$

In comparison with the initial formulation, the last summation in the objective function now takes into account in which period each operation takes place. The reason is that we may decide per workstation, and hence per period during the production lead time, how much will be produced in regular and how much in overtime. Naturally, regular production is cheaper, hence we attempt to produce as much as possible in regular time. If however, inventory-holding costs are relatively high, it may be advantageous to use the overtime option for production short in advance of needs, instead of producing earlier and pay more holding costs. The latter option is also more realistic if it is difficult to forecast periodic demand too long in advance.

The capacity and production planning problem can also be extended in another way. Consider the situation introduced in Sect. 21.1 (see Fig. 21.1) in which a final assembly department is preceded by a parts manufacturing shop. Then a logical procedure is to first plan the assembly department, using the Linear Programming formulation, followed by the planning of the parts manufacturing department. The periodic demand for parts $\bar{D}_{pt}$ follows from the just defined production plan of the final assembly department, as follows. Let the parts be indexed by p ($p = 1, \ldots, P$), and let b_{pj} denote the number of parts of type p that are assembled in one product j. Then the demand $\bar{D}_{pt}$ for parts p is defined by

$$\bar{D}_{pt} = \sum_{j=1}^{N} b_{pj} Q_{jt}, \quad p = 1, \ldots, P;\ t = 1, 2, \ldots$$

which in turn drives the parts manufacturing plan, using the off-set lead times $\bar{L}_p$ of parts $p (p = 1, \ldots, P)$. When linking the two models but solving them separately, the overall solution may still not be optimal in terms of overall costs. When formulating the integrated parts manufacturing—final assembly model, one immediately sees that the integrated objective function is the sum of the two objective functions in the separated (although linked) models. Since also the two constraint sets of the separated problems are not interfering, the overall minimum of the integrated problem is always smaller than the sum of the minima of the separated problems.

Several alternative formulations have been proposed by various authors, although the formulation above with the integration of product- and part-dependent lead times has to the best of our knowledge not been proposed earlier. Hopp and Spearman (2008) discuss an LP-formulation for a single-stage multi-product problem in which all orders are completely produced in the period in which they are released. They also allow for backorders, discuss overtime production and present a model for workforce planning in which costs are incurred with any change of staff level (hiring or firing), in this way forcing the management to keep the workforce as stable as possible.

Another issue is whether reliable forecasts on a product level be realistically made over a longer planning horizon. To that end, Hierarchical Production Planning (see also Chap. 12) offers an alternative by first planning capacity on an aggregate (product family) level in which a family consists of products with approximately the same capacity requirements, see Hax and Candea (1984), while later disaggregating this plan to production plans on an individual product level. Bitran et al. (1982) apply this approach for a two-stage manufacturing system similar to ours but focus on a single resource (manual labor) only while all lead times are equal.

Another way to reduce the need to generate demand forecasts that may turn out to be less reliable over a longer time period is to integrate the linear programming models in a rolling horizon framework in which forecasts are periodically updated. After solving the LP (or LP's in a 2-stage problem) only the first decisions are implemented, say to period $\tilde{T} < T - \max_j L_j - \max_p \bar{L}_p$ after which the time index is shifted (period $\tilde{T} + 1$ becomes period 1), and the whole procedure is repeated. Obviously, capacity profiles have to be updated carefully as well, taking into account the effects of decisions that have been taken in or shortly before period $\tilde{T}$ since they are affecting the free capacity of workstations also after (as a result of longer lead times) as well as the initial inventory positions but such procedures will not be discussed in detail here.

Once more, we wish to stipulate that the linear programming model basically serves to smooth capacity over time, making sure that expected demand will be met by balancing the capacities against inventories, possibly with additional options such as overtime work or outsourcing (the latter not discussed here in detail). The main idea is to cope with highly volatile, but predictable, demand patterns, to make sure that parts and products are ready when needed. They are by no means meant to deal with short term demand fluctuations for which stochastic models are more appropriate, both to model workstations and order routings (see the next section) as well as to determine safety stock levels (Chap. 20 of this volume). However, it is important that the lead times L_j used in this section realistically reflect the actual situation. It is well known that the off-set lead times used in e.g. MRP systems are basically larger than necessary, to cover all possible foreseen and unforeseen effects (batching, machine failures, quality problems). Besides, a high workload typically causes lead times to grow quite fast, leading to expediting work orders at the cost of others, etc. For that reason, workload control is often applied to keep lead times stable. How workload control is affecting the effective capacity (throughput) and the resulting lead times of a manufacturing shop is the topic of the next section.

19.4 Throughput and Lead Time Under Workload Control: A Queuing Network Analysis (Advanced)

To determine how the average lead time of a part of type k in the manufacturing shop is impacted by the load of the shop, we model the parts manufacturing department as a Closed Queueing Network (CQN) with multiple part types, see e.g. Chen and Yao

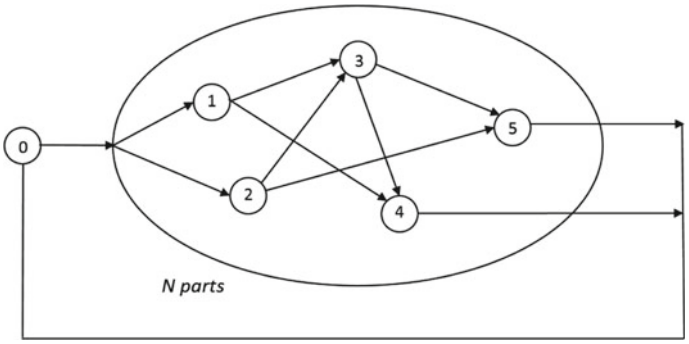

Fig. 19.2 Manufacturing shop with single machine workstations and a fixed number of jobs (parts)

(2001). The processing time of a part at workstation m is exponentially distributed with rate μ_m ($m = 1, \ldots, M$). At each workstation, parts are processed according to a FCFS (first-come, first-served) routine. Each part needs a card attached, at Station 0, to guide it through the system. Upon completion of its final operation, the job is returned to Station 0 where the card is detached, to be used for the next job. Because we are interested in the maximum production capacity of a system with say N cards, we assume that there are always jobs present to be loaded if a card is freed. Station 0 is called the load/unload station; we assume that loading and unloading time is negligible. Define routing probabilities as follows:

P_{0m} the probability that the first operation takes place at station $m, m = 0, 1, \ldots, M$.
P_{km} the probability that a job, after visiting station k, next visits station m, for $k, m = 0, 1, \ldots, M$.
P_{m0} the probability that a job, after being processed at station $m (m = 0, 1, \ldots, M)$, leaves the system.

The fact that both processing times and routings are probabilistic reflects a part type aggregation step; in fact, we work with a generic part, representing various underlying part types that may differ in processing time and routing. All assumptions can be relaxed, the current simple model is used as a starting point to present an important analysis method, called Mean Value Analysis (MVA), see Reiser and Lavenberg (1980). Clearly, the N cards are a means to implement a workload control rule, i.e., to allow a maximum of N jobs to be present simultaneously in the workshop. To determine the maximum system throughput (capacity), we therefore have to analyze a system that is always fully occupied, i.e., one in which all N cards are continuously circulating. For ease of presentation, we consider a very simple job shop first, in which each workstation consists of a single machine, cf. Fig. 19.2.

The question now is to determine the network production rate (throughput) of the aggregated parts as a function of N, as measured by the load/unload station and denoted by $TH_0(N)$, as well as the lead time $L(N)$ which is the time between the moment a part is loaded at station 0 and its return to that station after completion

of its final operation, to be unloaded. Before introducing the Mean Value Analysis (MVA) algorithm in its simplest form, let us introduce some notation:

V_m	visit ratio of station m, relative to the load/unload station 0.
$Q_m(n)$	expected number of parts present at station m, when there are n parts in the network.
$L_m(n)$	lead time experienced at station m for an arbitrary part, when there are n parts in the network.
$TH_m(n)$	throughput at station m, when there are n parts in the network

Next, we formulate the MVA algorithm (which is exact for this specific situation) to calculate the throughput $TH_0(N)$ and the lead time $L(N)$. In fact, we will iteratively determine the $TH_0(n)$ and lead time $L(n)$ for any integer value $n \leq N$, which will appear to be useful later.

Mean value analysis algorithm for parts of one type in simple job shops

1. *(Initialization) Set the visit ratio V_0 to station 0 equal to 1. Determine the other visit ratio's $V_m, m = 1, \ldots, M$, from the set of linear equations determined by the routing matrix, i.e.,*

$$V_m = \sum_{k=0}^{M} V_k P_{km}$$

   Set $n = 0$ and $Q_m(0) = 0, m = 0, \ldots, M$.
2. $n := n + 1$.
3. *Compute $L_m(n), m = 0, \ldots, M$, from*

$$L_m(n) = \big[Q_m(n-1) + 1\big]\frac{1}{\mu_m}$$

4. *Compute $L(n)$ and $TH_0(n)$ from*

$$L(n) = \sum_{m=0}^{M} V_m L_m(n)$$

$$TH_0(n) = \frac{n}{L(n)}$$

   and compute $TH_m(n), m = 1, \ldots, M$ from $TH_m(n) = V_m TH_0(n)$.
5. *Compute $Q_m(n), m = 0, \ldots, M$ from*

$$Q_m(n) = TH_m(n) L_m(n)$$

6. *If $n = N$ then STOP, else go to Step 2.*

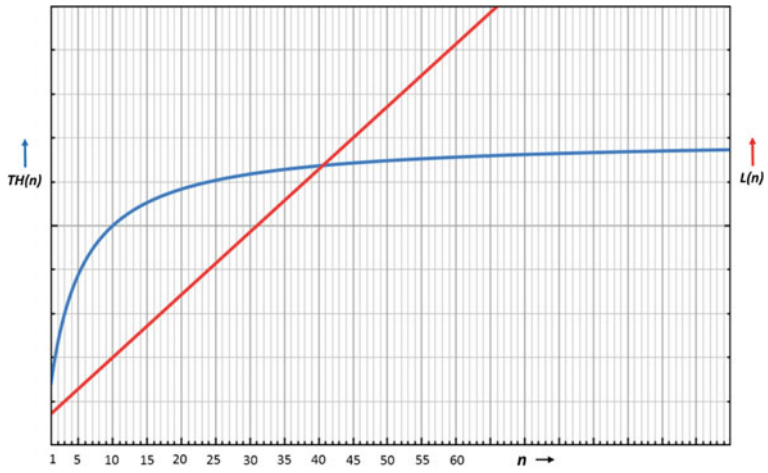

Fig. 19.3 Throughput and lead time as a function of the number of parts in the system

Note that the MVA algorithm computes the system production rate (throughput) for any possible number of parts in the system. When displaying the calculated throughputs and lead times graphically, one may observe some remarkable phenomena (cf. Fig. 19.3). First, the system throughput asymptotically approaches a limit, which may be detected as the capacity of the bottleneck machine in the system, i.e., the machine m with the largest value of $\frac{V_m}{\mu_m}$ (hence the machine that spends the most time to a single part, taking into account the visit ratio). Clearly, with a low number of parts in the system, even a bottleneck machine may be temporarily idle, but at the same time, loading too many jobs in the system does not improve system throughput any further. The system lead time increases asymptotically linear with the number of parts in the system, since $L(n) = \frac{n}{TH_0(n)}$. Hence, one may conclude that the system should be sufficiently loaded to attain an acceptable throughput but not more since that would only increase the system lead time. This fundamental observation is the basis of the workload control concept—loading too many jobs in the system only increases the lead time and does not help to generate a significantly higher throughput.

One may wonder why it is not possible in general to load the system such that exactly a bottleneck machine is always kept busy. For instance, a machine visited by every part exactly once, having an average processing time of 10 min, should be able to continuously work with an input flow of six jobs per hour. However, there are two reasons why this argument doesn't hold. First, it should require a deterministic interarrival time of exactly 10 min at the machine as well. The latter cannot be guaranteed because other stations may be visited by the part before it visits this particular machine. It is even more difficult to ensure a deterministic interarrival time if the visit schemes differ among parts queuing up in front of a bottleneck machine. Second, an average processing time of 10 min does not mean

that every processing time is exactly 10 min. Apart from the aggregation discussed earlier, processing times are seldom deterministic; they usually vary due to operator, machine or material interruptions, set-ups required, quality problems, etc. On the other hand, if the number of parts loaded to the system is sufficiently high, there will be almost always parts buffered in front of the machine, preventing idleness of the bottleneck.

The model and analysis presented above can be generalized in many ways. Workstations may contain multiple machines processing multiple part type families, each with their own (probabilistic) routing and processing time characteristics, including arbitrarily distributed processing times. In addition, extensions exist to include batching and set-up times (adding more variance to processing times, because typically the first part in a batch includes the set-up time in its process time), and quality problems causing possible rework loops (again a source of variability). Under these more general characteristics, the analysis is no longer exact but the procedures developed have been extensively tested and have proven to provide near-accurate estimates of important performance measures. The generalized MVA algorithms are mathematically complex and therefore beyond the scope of this chapter. Important however is that the general picture remains the same. The throughput increases with the number of parts loaded, but the function is concave. In other words, the law of diminishing returns applies. Beyond a threshold, additional parts loaded do not increase the throughput, instead lead to significant longer lead times.

19.5 Workload Control Under External Demand for Production to Stock (*Advanced*)

So far, note that the CQN model is only used to determine the *maximum* capacity of a manufacturing department under a workload control regime (i.e., a regime that limits the number of parts released to the cell), consisting of various work stations that may be visited by various part types, each having their own routing through the cell. To that end, it was natural to assume that the department was always fully loaded. The parts manufacturing shop discussed in the introduction of this chapter however is subject to demand generated from a final assembly schedule. Because the variety of part types was limited in comparison to the large variety of end-items, parts are produced to stock (the CODP is located between parts manufacturing and assembly) to ensure that assembly is never idle due to a lack of parts. At the same time, a workload control regime limits the number of parts to be simultaneously processed to some number N. The question is how the preceding analysis can be used to accurately determine all performance measures of interest, including the fill rate of the finished parts inventories, the parts manufacturing lead times, and of course the resulting effective throughputs.

In this section, we therefore consider a parts manufacturing department that produces items to be placed in stock, in anticipation of their use in a subsequent assembly

phase. The parts manufacturing department is subject to a workload control regime. See Fig. 19.4, in which we restrict ourselves to one (generic) part type again to simplify explanation.

In this system two control parameters play a crucial role, the predetermined stock level of finished parts S (called the base stock level) and the number of cards N. For each job (a part to be processed in the manufacturing shop), a card needs to be attached to guide it through the system. Hence, the number of cards N limits the number of jobs to be processed simultaneously. However, contrary to the capacity analysis in the preceding section, here not all cards are always in use; if demand from assembly shows a temporary decline, the production requirements diminish and hence a number of cards may be temporary idle. Now let

$m =$ the number of finished parts placed in stock, $m \leq S$,
$k =$ the number of backordered demands for finished parts,
$a =$ the number of backordered production requests for parts,
$b =$ the number of free Kanbans available for parts production, $b \leq N$.

If a request for a part arrives from the assembly department, it is split into two sub-requests. The first one is directed to the stock of finished parts m, at synchronization station J_s. If a product is available in stock, it is used to satisfy demand, otherwise it joins a queue (k) of requests still waiting to be fulfilled (demand is backordered). The second sub-request reflects a production order for a similar part in order to make sure that the inventory position of finished parts is eventually replenished. However, the production order is released if and only if it is authorized by a card from the queue of free cards (b) at synchronization station J_c. If no free card is available, the production order joins a queue of orders that are waiting for a card and hence still

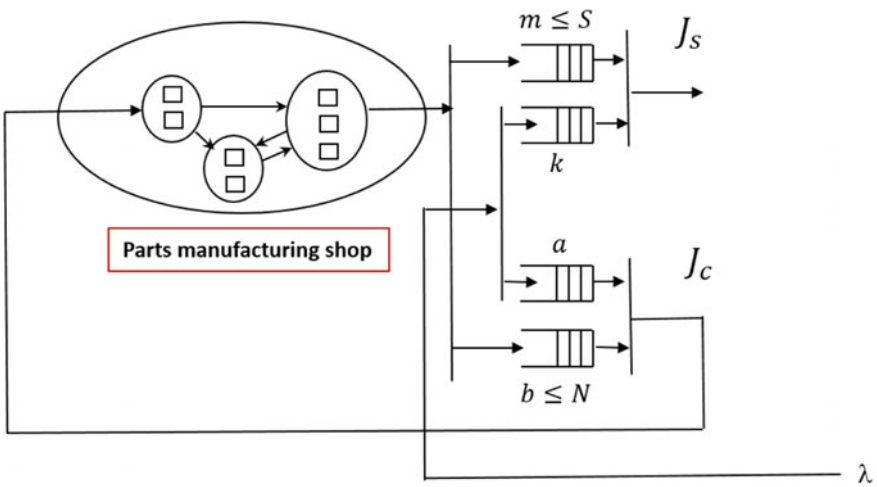

Fig. 19.4 A production-to-stock parts manufacturing shop controlled by a generalized Kanban control (GKC) system

have to be released. Finally, let $\vec{n}$ denote the vector with each element the number of parts at a workstation in the parts manufacturing department. Then, it is easily seen that:

$$
\begin{aligned}
|\vec{n}| + a + m - k &= S \\
|\vec{n}| + b &= N \\
m \cdot k &= 0 \\
a \cdot b &= 0
\end{aligned}
$$

Subtracting the first equation from the second equation and rearranging terms leads to

$$(m - k) = (b - a) + (S - N)$$

Note that, if $b - a$ is known, we know both b and a separately because both variables are nonnegative and cannot be non-zero simultaneously. If $b - a = c$ then $c \geq 0$ implies $b = c$ and $a = 0$, while $c < 0$ implies $b = 0$ and $a = -c$. The same holds for $m - k$, in other words, both synchronization stations constitute a one-dimensional queueing system, with states described by $b - a$ and $m - k$ respectively, which differ by a value $S - N$. Both queues act as a one-dimensional birth and death queue of which the state is increased by one due to a parts job completion, and is decreased by one due to an arrival of a finished parts request.

Also observe that the parts manufacturing shop, together with synchronization station J_c, is in fact very similar to the closed shop in the preceding section, where station J_c takes the role of the load/unload station. The only, important, difference is that now a card that is dismissed after completion of a parts manufacturing job, is not immediately coupled to a new part to enter the shop, but may have to wait until all free cards ahead of it are occupied and then finally is attached to the next arrival for a parts production request. Suppose for the moment that the arrival of parts production requests is governed by a Poisson process with rate λ. Let $c = b - a$. Then the transition diagram between the states c (i.e., the birth and death process) is displayed as in Fig. 19.5, where $TH(n)$ is the throughput of the parts manufacturing department treated as a CQN n cards circulating through the shop (as determined in the preceding section).

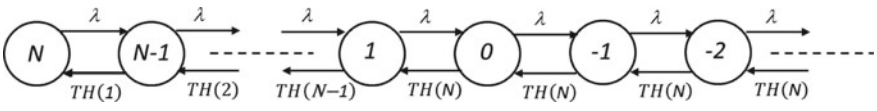

Fig. 19.5 Transition diagram of synchronization station J_c

We now assume that the throughput rates $TH(n)$ are exponentially and independently distributed (which is not true in general). This allows us to determine the probabilities $p(c)$, as follows:

$$p(c) = \left(\frac{\lambda^{N-c}}{\prod_{j=1}^{N-c} TH(j)} \right) p(N), \quad c = N-1, N-2, \ldots, 0,$$

$$p(c) = \left(\frac{\lambda}{TH(N)} \right)^{-c} \left(\frac{\lambda^{N}}{\prod_{j=1}^{N} TH(j)} \right) p(N), \quad c = -1, -2, \ldots$$

Finally, $p(N)$ follows as a normalization constant, i.e., from $\sum_{c=-\infty}^{N} p(c) = 1$.

The number of parts n actually in process in the shop is now easily determined from

$$n = N - \max(c, 0)$$

while also the probability distribution of $m - k$, and hence of m and k separately follows immediately from the probability distribution of $c = b - a$. From this, all relevant performance indices are now detected. As an example, we present the fill rate $FR(S)$ of the stock of finished parts, i.e., the probability that an arriving request for a finished part is immediately fulfilled, when the base stock level is equal to S. We have

$$FR(S) = \sum_{m=1}^{S} p(m) = \sum_{c=N+1-S}^{N} p(c)$$

while for the expected time to fill a part request $ETFPR(S)$, the following equation holds.

$$ETFPR(S) = \sum_{k=0}^{\infty} (k+1)p(k)/\lambda = \sum_{c=-\infty}^{N-S} (N - S - c + 1)p(c)/\lambda$$

The average lead time $L^*(N)$ in the parts manufacturing shop satisfies

$$L^*(N) = L(N) + \sum_{a=1}^{\infty} ap(a) = L(N) + \sum_{c=-\infty}^{-1} (-c)p(c)$$

where the first term on the right-hand side is the lead time as determined for a maximally loaded shop (as in Sect. 19.3) and the second term denotes the average waiting time before a card becomes available and hence production can be started.

The above analysis can be extended to multiple part type families, each with their own routing. The analysis is dependent on whether one general workload norm for all part type families simultaneously holds (meaning one set of cards that limits the

overall workload) or a specific workload per part type family is specified (meaning a specific set of cards for each part type family). The analysis is beyond the scope of this chapter; we will not discuss it here any further.

Further extensions to multi-stage systems in which each department is subject to a workload control system are also possible. Instead of an external demand modeled by a Poisson process with rate, we then encounter a state-dependent external demand rate (demand depends on the workload control rule, i.e., the number of cards, and the base-stock levels of a subsequent assembly section, whereas material availability in front of the parts manufacturing department again depends on base-stock levels and possibly a workload control rule at a preceding stage. We refer to the literature at the end of this chapter for further reading.

Recall that in the entire analysis in this section we have assumed an FCFS priority discipline at each workstation, based upon which lead times have been determined. One may wonder whether at a detailed operational level smart scheduling systems might help to meet the due dates induced by these pre-determined lead times. That will be discussed in the final section.

19.6 Job Shop Scheduling (*State-of-the-Art*)

In the preceding section, we determined the relation between the effective capacity and the expected lead time in the parts manufacturing shop. Based upon these expected lead times, we might set a due-date for each individual shop at the time of actual release to the shop floor. However, note that these due dates are then based on an *average* lead times and on a first-come-first-served priority discipline at individual workstations. In the current section we investigate whether at an operational level a more smart scheduling discipline can help to ensure that as many jobs as possible will indeed meet their scheduled due date. Such an advanced scheduling discipline exists. In this section, we outline the basic principles of what is known as the Shifting Bottleneck heuristic for a relatively simple job shop. Similar to the procedures in the previous section, also this heuristic can be extended rather easily to cope with generic versions of the job shop scheduling problem and is therefore suitable to be used in practice where all kinds of additional constraints may exist, both with respect to job as well as equipment constraints.

Below, we first provide a formal description of the job shop scheduling problem that we need to solve. We then show how such problems can be represented by a disjunctive graph, after which we discus show to use the concept of selection to specify solutions for the job shop scheduling problem.

Problem description

In the job shop scheduling problem in its simplest version, a finite set of jobs $J = \{1, \ldots, n\}$ needs to be scheduled on M machines, numbered $1, 2, \ldots, M$. Each job $j \in J$ consists of a number of operations that need to be processed in a specified sequence. For the sake of notational convenience, we assume that each job consists

of M operations and that each job visits every machine exactly once, i.e., each job has one operation that needs to be processed on machine m ($m = 1, \ldots, M$). Note that the sequences in which the jobs have to visit the machines may differ from job to job. Let O_{mj} be the operation of job j that needs to be processed on machine m and let t_{mj} denote the processing time. Moreover, let d_j be the due date of job j, i.e., the time at which job j should be finished (as determined by its release time and the lead time as determined in the preceding section).

Each machine can only process one operation at a time and once a machine starts processing an operation, it needs to finish processing this operation before it can start processing a next operation. Therefore, for each machine, we should find a sequence in which it processes its operations. A *schedule* σ specifies for each operation O_{mj} its start time $S_{mj}(\sigma)$ (i.e., the time at which machine m starts processing operation O_{mj}) and its completion time $C_{mj}(\sigma) = S_{mj}(\sigma) + t_{mj}$ (i.e., the time at which machine m completes operation O_{mj}). The *maximum lateness* of schedule σ is defined as

$$L_{\max}(\sigma) = \max_{m,j} L_{mj}(\sigma) = \max_{m,j} \{C_{mj}(\sigma) - d_j\}$$

The objective is to find a schedule σ^* in which the maximum lateness $L_{\max}(\sigma^*)$ is as small as possible, which means that $L_{\max}(\sigma^*) = \min_{\sigma} L_{\max}(\sigma)$.

Disjunctive graph representation

Each instance of the job shop scheduling problem can be represented by a graph $G = (V, A)$, with V a set of nodes and A a set of arcs. V consists of a node v_{mj} for each operation O_{mj} and two dummy nodes, S_1 (the *source*) and S_2 (the *sink*). The weight of node v_{mj} is equal to the processing time t_{mj} of operation O_{mj}. The weights of the nodes S_1 and S_2 are equal to 0.

Suppose that O_{hj} and O_{mj} are two consecutive operations of job j, which means that when job j has been processed on machine h, it needs to be processed next on machine m. For each pair of consecutive operations O_{hj} and O_{mj}, A contains an arc (v_{hj}, v_{mj}). Moreover, A contains an arc (S_1, v_{mj}) if operation O_{mj} is the first operation of job j. This means that there are n such arcs in total (one for each job). Finally, A contains n directed arcs from the last operation of each job to S_2. Together, these arcs form the *conjunctive arcs* of G. These conjunctive arcs represent the sequence in which the operations of each job need to be processed. Let C be the set consisting of these conjunctive arcs.

Apart from the conjunctive arcs, A contains *disjunctive arcs* as well. These disjunctive arcs represent the machine conflicts. Because each machine can process at most one operation at a time, in a solution for the job shop scheduling problem we need to specify the sequences in which the machines process their operations. For every pair of operations O_{mj} and O_{mk} that need to be processed on the same machine m, A contains a pair of arcs, namely (v_{mj}, v_{mk}) and (v_{mk}, v_{mj}), so one arc in both directions.

Each arc has a weight of 0, except for the arcs to S_2. To model the job due dates, the arc from the node representing the last operation of job j to S_2 gets a weight

Table 19.1 Data of an example instance

Job$j$	d_j	$O_{mj}(t_{mj})$		
1	17	O_{11} (4)	O_{21} (7)	O_{31} (6)
2	18	O_{32} (3)	O_{12} (5)	O_{22} (8)
3	14	O_{23} (2)	O_{33} (6)	O_{13} (7)

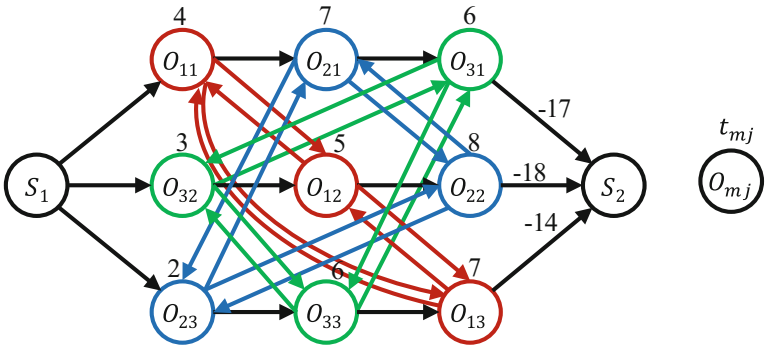

Fig. 19.6 Disjunctive graph representation

equal to $-d_j$. Figure 19.6 shows the disjunctive graph representation of a job shop scheduling instance shown in Table 19.1.

Selection

Now we have represented the job shop scheduling problem by a disjunctive graph. In this subsection, we subsequently show how a solution can be specified using this graph.

Recall that the graph G contains a pair of arcs (one in each direction) between nodes v_{mj} and v_{mk}, representing two operations that need to be processed on the same machine m. A *selection* is a set of arcs that contains exactly one arc of each of these arc pairs. Let $G_S = (V, C \cup S)$ be the graph that has the same node set V as the disjunctive graph G; moreover, the arc set of G_S consists of the conjunctive arc set C of G plus the selection S. A selection S is called *feasible* if the corresponding graph G_S is *acyclic*, i.e. if it does not contain a directed cycle.

Recall that the objective in the job shop scheduling problem is to minimize the maximum lateness $L_{\max}(\sigma)$. This objective function is a *regular* objective function, which means that, given the sequences of operations on machines, it is always best to start each operation as early as possible. So, given a feasible selection S, we can determine its associated schedule σ_S by determining the earliest possible operation start and completion times, given the sequences of operations on machines implied by selection S. Figure 21.7 shows the graph G_S for a feasible selection S and its associated schedule.

Given a feasible selection S, the schedule σ_S associated with it can be calculated as follows. Let $l_{mj}(S)$ be the length of a longest path from S_1 to v_{mj} in the graph G_S. The length $l_{mj}(S)$ is defined as the sum of the weights of the nodes and arcs that are part of the longest path from S_1 to v_{mj} in graph G_S, excluding the weight of node v_{mj}.

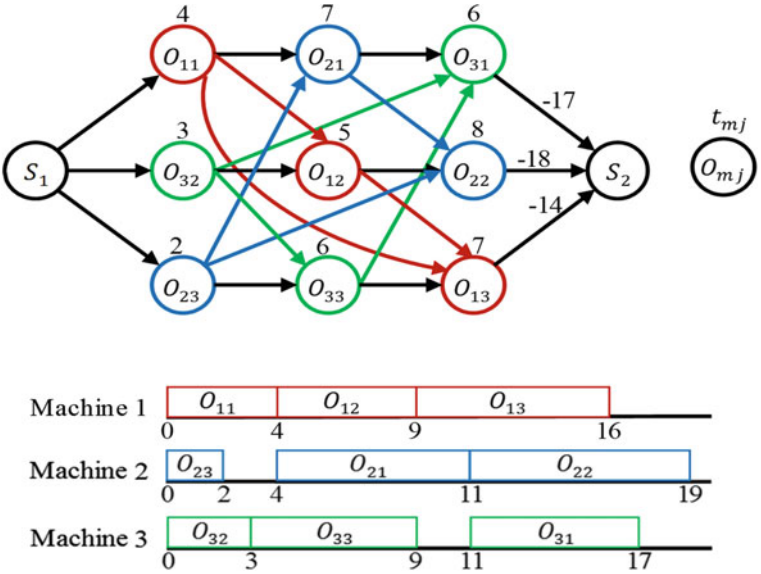

Fig. 19.7 Graph G_S and its associated schedule

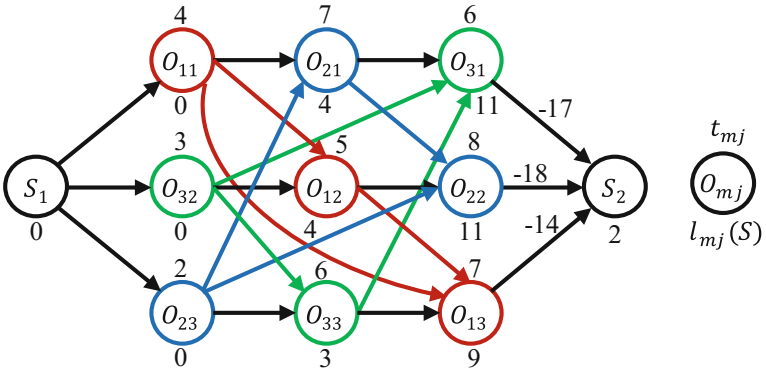

Fig. 19.8 Longest path calculations in graph G_S

Given a feasible selection S, the earliest possible starting time of operation O_{mj} is equal to $l_{mj}(S)$ (assuming that we begin the scheduling procedure at time 0, e.g. at the beginning of each week). This means that $S_{mj}(\sigma_S) = l_{mj}(S)$. Moreover, $L_{\max}(\sigma_S)$ is equal to the length of a longest path from S_1 to S_2. Figure 19.8 shows the longest path calculations for the same selection S as shown in Fig. 19.7 (resulting in the start times of all operations and in the maximum lateness $L_{\max}(\sigma_S)$).

It is not difficult to see that there exists a selection S^* that results in an optimal schedule σ^*. Finding an optimal schedule (specifying the start and completion times of all operations) is therefore equal to finding the best selection (specifying the

operation sequences on machines, based on which the operation start and completion times can be calculated).

19.7 The Shifting Bottleneck Heuristic (*State-of-the-Art*)

The Shifting Bottleneck (SB) is an iterative heuristic that decomposes the job shop scheduling problem into single-machine scheduling problems. In each iteration, one additional machine is scheduled until all machines are scheduled; the result is then a schedule for the job shop scheduling problem. In this section, we first discuss the Shifting Bottleneck heuristic and then zoom in on the required steps.

The idea behind the SB heuristic is to focus first on machines that have most impact on the job shop schedule. The machine that is identified as having the most impact is called the first bottleneck machine and a schedule for this machine is determined. Next, in the second iteration, the consequences of the schedule of the first bottleneck machine is determined for each of the remaining machines. Then, of the remaining (non-bottleneck) machines, the machine with the most impact is determined. This machine is the next bottleneck machine and a schedule for this machine is fixed. At the end of the second iteration, the impact is determined that the schedule of second bottleneck machine has on the schedule of the first bottleneck machine. Possibly, the schedule of the first bottleneck machine is adapted to get a better overall result. This latter step is called the *bottleneck reoptimization step* and it ends the second iteration of the SB heuristic. In the third iteration, the third bottleneck machine is identified, its schedule is determined, and the bottleneck reoptimization step is performed. After the Mth iteration, the SB heuristic terminates. Before discussing the technical details of the steps, we first present a pseudocode for the steps at a high level:

- *//Initialization*
- $\mathcal{M} := \emptyset$
- *//Execute M iterations to schedule all machines*
- For $m := 1$ to M do
  - Find next bottleneck machine (b_m)
  - Fix schedule of new bottleneck machine b_m
  - Reoptimize the $(m-1)$ existing bottleneck machines
  - $\mathcal{M} := \mathcal{M} \cup \{b_m\}$

We now proceed to discuss the technical details of the steps in the Shifting Bottleneck heuristic.

Find next bottleneck machine and its schedule

At the start of iteration m of the SB heuristic, the schedules of $(m-1)$ bottleneck machines have been fixed. For these machines, we therefore have sequences in which these machines process their operations. In other words, we have a *partial* selection S'. Consider now the graph $G_{S'}$. If we ignore the capacity constraints of non-bottleneck machines, $r_{mj}(S') := l_{mj}(S')$ gives the earliest possible starting time

(i.e., the release date) of each operation O_{mj}, while the length of a longest path in $G_{S'}$ from S_1 to S_2 gives the maximum lateness $L_{\max}(\sigma_{S'})$. Let $l'_{mj}(S')$ be the length of a longest path in $G_{S'}$ from v_{mj} to S_2. If we do not want to increase the current maximum lateness $L_{\max}(\sigma_{S'})$, then operation O_{mj} should start no later than $L_{\max}(\sigma_{S'}) - l'_{mj}(S')$. Therefore, $d_{mj}(S') := L_{\max}(\sigma_{S'}) - l'_{mj}(S') + t_{mj}$ is a due date for operation O_{mj}.

Based on longest path calculations in $G_{S'}$, we have an earliest possible starting time $r_{mj}(S')$ and a due date $d_{mj}(S')$ for each operation O_{mj}. For each of the non-bottleneck machines, we now determine the optimal (single-machine) schedule, taking into account the release and due dates by means by an algorithm derived by Carlier (1982); the objective in these single-machine scheduling problems is also to minimize the maximum lateness.

The non-bottleneck machine that has the highest maximum lateness is the next bottleneck machine. The schedule that is fixed for this machine is the schedule that results in the optimal maximum lateness.

Reoptimize the existing bottleneck machines

Suppose again that we are at the mth iteration of the SB heuristic and, moreover, that we just identified the mth bottleneck machine. To finalize the mth iteration of the heuristic, we now perform the bottleneck reoptimization step. Suppose that b_k is the bottleneck machine found in the kth iteration ($k = 1, \ldots, m$). During the bottleneck reoptimization step, the $m - 1$ existing bottleneck machines are rescheduled *one by one*, taking into account the schedules on the other bottleneck machines, including the one identified in this (mth) iteration. Once an existing bottleneck machine is rescheduled, the newly found schedule replaces the existing schedule for this bottleneck machine.

To find a new schedule for bottleneck machine b_k, consider the partial selection S'_k that consists of the newly found schedules for machines $b_1, b_2, \ldots, b_{k-1}$ and the schedules for machines $b_{k+1}, b_{k+2}, \ldots, b_m$. For operations $O_{b_k,j}$ on bottleneck machine b_k, the release and due dates are calculated in the graph $G_{S'_k}$, again based on longest path calculations. The new schedule for machine b_k is the schedule on this machine that minimizes the maximum lateness given the release and due dates.
In pseudocode, the bottleneck reoptimization procedure reads as follows:

- *//at the start, we have the existing bottleneck*
- *//machines* $b_1, b_2, \ldots, b_{m-1}$ *and*
- *//a new bottleneck machine* b_m
- For $k := 1$ to $m - 1$ do
  - Construct $G_{S'_k}$
  - Calculate release and due dates for operations on machine b_k
  - Find optimal schedule for machine b_k.

Practical extensions

One of the strengths of the SB heuristic is the possibility to modify it such that it can be applied to far more general versions of the job shop scheduling encountered in practice. The modifications typically consist of (a combination of) changes in the

weights of the nodes and arcs in the disjunctive graph and using adapted algorithms to solve the single-stage scheduling problems (which are single machine scheduling problems in the version described above). In this way, the SB heuristic is for example able to deal with setup times on machines, or workstations that consist of parallel identical machines instead of a single machine.

19.8 Further Reading

In this chapter, we have presented algorithms for executing important building blocks of the framework presented in Chap. 5 and present the basic structure of a (metal working) company as often exists in practice. Such a company is detailed in the case study presented in Sect. 21.2, which is derived from Slomp (1993). The linear programming formulations discussed in Sect. 21.3 uses elements from various publications, including Silver et al. (2017), Bitran et al. (1982) and Hopp and Spearman (2008). Interesting also are attempts to integrate MRP and HPP, see e.g. Hax and Meal (1975) and Hax and Candea (1984).

The relation between workload, throughput (effective capacity) and internal lead times in a manufacturing shop are discussed by Hopp and Spearman (2008), but the complete technical basis is due to Buzacott and Shanthikumar (1993), using a queueing network approach. This work (and the papers it is based on) led to a rich offspring, see e.g. Dallery (1990), Di Mascolo et al. (1996), and Buitenhek et al. (2000), while an attempt to place the work in a broader manufacturing planning and control context was presented by Zijm (2000). Zijm and Buitenhek (1996) investigate the integration of a due-date setting approach based on (open) queuing networks and the subsequent application of a Shifting Bottleneck heuristic.

Job Shop scheduling is a topic that has received much attention in combinatorial optimization of which the book by Pinedo and Chao (1996) presents a nice review, see also Brucker et al. (1994). The Shifting Bottleneck procedure was introduced by Adams et al. (1988), using Carlier's one-machine scheduling problem as a building block (Carlier 1982), and generated quite some offspring, see e.g. Ivens and Lambrecht (1996). Extensions that make the procedure applicable to more realistic machining systems are due to Schutten (1998), Schutten and Leussink (1996) and Schutten et al. (1996).

The integration of workload control, lead time off-setting and shop floor scheduling remains a challenging task. The wealth of generic heuristics for combinatorial optimization problems (simulated annealing, taboo search, genetic algorithms) have contributed significantly to deriving close-to-optimal solutions for these problems. At the same time, they do not provide the insights that can be expected from a further integration of queueing and smart scheduling approaches. The latter defines a rich field for future research.

References

Adams J, Balas E, Zawack D (1988) The shifting bottleneck procedure for job shop scheduling. Manag Sci 34:391–401

Bitran GR, Haas EA, Hax AC (1982) Hierarchical production planning: a two stage system. Oper Res 30:232–251

Brucker P, Jurisch B, Sievers B (1994) A branch and bound algorithm for the job-shop scheduling problem. Discrete Appl Math 49:107–127

Buitenhek R, van Houtum GJ, Zijm WHM (2000) AMVA based solution procedures for open queueing networks with a population constraint. Ann Oper Res 93:15–40

Buzacott JA, Shanthikumar JG (1993) Stochastic models of manufacturing systems. Prentice-Hall, Englewood Cliffs, NJ

Carlier J (1982) The one-machine sequencing problem. Eur J Oper Res 11:42–47

Chen H, Yao DD (2001) Fundamentals of queueing networks—performance, asymptotics and optimization. Springer, New York

Dallery Y (1990) Approximate analysis of general open queueing networks with restricted capacity. Perform Eval 11:209–222

Di Mascolo M, Frein Y, Dallery Y (1996) An analytical method for performance evaluation of Kanban-controlled production systems. Oper Res 44(1):50–64

Hax AC, Candea D (1984) Production and inventory management. Prentice-Hall, Englewood Cliffs, NJ

Hax AC, Meal HC (1975) Hierarchical integration of production planning and scheduling. In: Geisler MA (ed) Logistics. Studies in the management sciences, vol 1. Elsevier, North-Holland

Hopp WJ, Spearman ML (2008) Factory physics: foundations of manufacturing management. Richard D. Irwin, Homewood, IL

Ivens P, Lambrecht M (1996) Extending the shifting bottleneck procedure to real-life applications. Eur J Oper Res 90:252–268

Pinedo M, Chao X (1996) Operations scheduling with applications to manufacturing and services. Irwin/McGraw-Hill, Boston

Reiser M, Lavenberg SS (1980) Mean-value analysis of closed multichain queueing networks. J Assoc Comput Mach 27:313—322

Silver EA, Pyke DF, Thomas DJ (2017) Inventory and production management in supply chains. CRC Press (Taylor and Francis), Boca Raton

Schutten JMJ (1998) Practical job shop scheduling. Ann Oper Res 83:161–177

Schutten JMJ, Leussink RAM (1996) Parallel machine scheduling with release dates, due dates and family setup times. Int J Prod Econ 46–47:119–125

Schutten JMJ, van de Velde SL, Zijm WHM (1996) Single-machine scheduling with release dates, due dates and family setup times. Manag Sci 42(8):1165–1174

Slomp J (1993) Production control for flexible manufacturing system—an application-oriented approach. PhD thesis, University of Twente

Zijm WHM, Buitenhek R (1996) Capacity planning and lead time management. Int J Prod Econ 46–47:165–179

Zijm WHM (2000) Towards intelligent manufacturing planning and control systems. OR Spectr 22:313–345