

Interaction of design and operational parameters in periodic review kanban systems

FERYAL ERHUN†, M. SELIM AKTURK‡* and AYTEN TURKCAN‡

In this study, we propose an analytical model to determine the withdrawal cycle length, kanban sizes and number of kanbans simultaneously in a multi-item, multi-stage, multi-period, capacitated periodic review kanban system. Traditionally, research in kanban systems has been separated into ‘design level’ research, and ‘shop floor level’ research. In both veins of research, especially for the design level, various algorithms have been developed. However, it is important to state that there is a significant relationship between the design parameters and the operational issues, such as kanban schedules and actual lead times. Therefore, we analytically consider the interdependencies between the design and operational decisions, and evaluate their impact on each other.

1. Introduction

Just-in-time (JIT) is a management philosophy that can be defined as the ideal of having the necessary amount of material available where it is needed and when it is needed. One of the major elements of JIT philosophy and pull mechanism is the kanban system. Kanban system is the information processing and hence shop floor control system of JIT philosophy. The system effectively limits the amount of in-process inventories and it coordinates production and transportation of consecutive stages of production in assembly-like fashion. In this study, we propose an analytical approach to determine the design parameters of the withdrawal cycle length, number of kanbans and kanban sizes simultaneously in a multi-item, multi-period, multi-stage, capacitated periodic review kanban system. Traditionally, research in kanban systems has been separated into ‘design level’ research and ‘shop floor level’ research. In both veins of research, especially for the design level, various algorithms have been developed. However, it is important to state that there is a significant relationship between the design parameters and the operational issues, such as kanban schedules and actual lead times. Therefore, we should consider the interdependencies between the design and operational decisions, and evaluate their impact on each other. Our objective is to integrate design and operational level decisions to bridge this gap. This new approach recognized an important tradeoff in JIT systems that has been largely unrecognized, and which we believe can be effectively exploited to improve production efficiency, and to lead substantial cost reductions.

Revision received February 2003.

† Management Science and Engineering, Stanford University, Stanford, CA 94305, USA.

‡ Department of Industrial Engineering, Bilkent University, 06800 Bilkent, Ankara, Turkey.

* To whom correspondence should be addressed. email: akturk@bilkent.edu.tr

For each part in the kanban system two basic design parameters have to be determined: the kanban size and the number of kanbans to be used. For a detailed literature review on model characteristics, refer to Akturk and Erhun (1999). The analytical models in current literature in general have several limitations. Almost all the models assume that the kanban sizes are known and the number of kanbans can be determined by using these predetermined values, material handling is instantaneous, and no sequencing decision is necessary. In fact, Karmarkar and Kekre (1989) have shown that there is a significant relationship between kanban sizes and production lead times, and hence the shop congestion. Therefore, the number of kanbans and kanban sizes should be determined simultaneously, as these two together affect the performance of the system. Philipoom *et al.* (1996) provide a nonlinear integer mathematical model for the multi-item, multi-stage, multi-period, capacitated kanban system. The model determines kanban sizes, number of kanbans, and final assembly sequence simultaneously by minimizing the set-up and inventory holding costs. Chang and Yih (1994b) use a simulated annealing approach to determine the number of kanbans and kanban sizes with the objectives of minimizing the cycle time, operating cost and work-in-process capital loss. Sarker and Balan (1999) find the number of kanbans circulated between workstations, finished good shipments to the buyers and the batch size for each shipment for a multi-stage production line where the demand of finished goods at the last stage increases linearly with time and no shortages allowed at any stage in the system.

Almost none of the models, except Philipoom *et al.* (1996), consider the impact of operational issues on design parameters. It is usually assumed that the control periods are small enough to ignore batch sequencing problems. Even in the study of Philipoom *et al.*, only the final assembly sequence is determined and it is assumed that it propagates back by the first come first served (FCFS) rule. The instantaneous kanban withdrawal mechanism is a typical assumption in most of the existing studies. On the other hand, there are a limited number of studies on periodic review systems. In periodic review kanban systems, the needed products are withdrawn periodically from the preceding stage. The period between the two consecutive material handling operations is denoted as withdrawal cycle length. Savsar (1996) has shown that kanban withdrawal policies might affect throughput rate and inventory levels of JIT production control systems, therefore the review frequency should be considered during the design of a kanban system. The problem of production leveling through scheduling is crucial in kanban systems. Selecting the proper scheduling rule becomes even more important in the case of imperfect production settings, in which demand may be variable, set-up times may be significant, and production line may be unbalanced (Huang and Kusiak 1996). Chang and Yih (1994a) introduce the concept of generic kanban systems in which the demand and the processing times are variable. Chang and Yih (1998) propose a fuzzy rule-based approach in order to determine the number of kanbans and kanban sizes in a generic kanban system, which change dynamically according to the demand realizations during the planning horizon.

Several models in the literature assume station independence through either assuming infinite capacity or not allowing back-orders as summarized in Akturk and Erhun (1999). Under JIT system, all stages are integrally tied to each other and if there is a delay in one stage then both the preceding and succeeding stages should be affected to highlight interdependencies among the stages. This is crucial for reducing the size of the inventory levels and eliminating overall waste. However,

this possibility is eliminated with the assumption of no back-orders and an infinite production capacity, hence each stage can be modeled independently. Furthermore, most of the conjectures for the determination of design parameters are posed (although not explicitly stated) as if these conjectures are independent of the operational decisions. For example, it is a common conjecture that the customer service level, usually expressed in terms of back-order costs and fill rates, can be increased by increasing the maximum inventory level, or the safety stock parameter in the Toyota formula. We will show that this conjecture is only valid for the FCFS rule, which maintains consistent kanban priorities from stage to stage and prevents kanban passing. For other scheduling rules, the best inventory level that maximizes the customer service level may not correspond to the highest possible inventory level.

There are a limited number of studies on determining the kanban sequences. In general, the final assembly schedule determines production schedules for all of the stages in the facility. Once the assembly line is scheduled, it is assumed that the sequence propagates back through the system using the FCFS rule. However, many researchers based on their simulation studies showed that the commonly adopted policy of the FCFS rule may not be a good policy under different experimental conditions, but in their models, they still limited themselves to another relatively simple dispatching rule of SPT/LATE. In a recent study, Hum and Lee (1998) consider a six-station single-card kanban-controlled line and provide a simulation evaluation of the performance of four scheduling rules, including FCFS and shortest processing time (SPT), operating under different production scenarios. In their simulation experiments, the FCFS rule performed poorly compared to the other dispatching rules especially under tight production conditions. This is similar to the previous results of Lee *et al.* (1993) and Yavuz and Satir (1995) where SPT/LATE (or SPT) was stated as the best overall rule.

Sequencing in kanban-controlled shops are more complex when compared to conventional sequencing problems as kanbans do not have due dates and kanban-controlled shops have station blocking (Berkley 1992). Station blocking can be described as the idleness of a stage due to full outbound inventories. Therefore, we will show that using a more sophisticated scheduling algorithm instead of the simple dispatching rules can significantly enhance kanban system performance. Furthermore, it is commonly assumed that the final assembly schedule can be propagated back through system using the FCFS rule with the instantaneous kanban withdrawal mechanism. As discussed above, this combination may not be a good policy for all experimental conditions. On the contrary, the FCFS rule performs better when the withdrawal cycle lengths are long enough to justify the set-up times under the tight production conditions. We will also show that the scheduling policies are not robust to withdrawal cycle lengths, or vice versa. The withdrawal cycle length has a significant effect on average in-process inventories and production rates. As the withdrawal cycle lengths decrease, the in-process inventories decrease significantly, but the back-orders increase. In fact, when the kanban sizes are constant, the total cost curve is convex over withdrawal cycle lengths. A detailed review on JIT and kanban systems can be found in Akturk and Erhun (1999), Berkley (1992), and Huang and Kusiak (1996).

The remainder of this paper is organized as follows. In section 2, we discuss the underlying assumptions and define the problem. We present the proposed algorithm in section 3 along with a numerical example in section 4. A computational analysis of

the proposed algorithm is reported in section 5. Finally, some concluding remarks are provided in section 6.

2. Problem statement

In this study, we develop an analytical model to determine the withdrawal cycle length, number of kanbans and kanban sizes simultaneously in a multi-item, multi-stage, multi-period, capacitated, periodic review kanban system. The overall objective is to minimize the total production cost, which is comprised of back-order and inventory holding costs, over a finite planning horizon. The existing studies reflect the relationship between the kanban sizes and the average inventory, but for the other performance measures no clear relations are present. One of the purposes of this study is to investigate analytically the effect of the kanban sizes on several system performance measures, such as average in-process inventory and total back-order cost.

The following assumptions are made throughout this study. Kanban processing times are equal to the time required to process all parts in a kanban. Half full kanbans are not allowed. There are no alternative routings. The withdrawal lead time is assumed to be zero. The system is reliable and production at each stage is carried out without defects. There is a discrete and time-varying demand pattern for every item, which comes from a given probability mass function. There are several assumptions necessary for kanban system to work efficiently and effectively. The plant layout must be flow based in order to reduce transportation times and highlight interdependencies between work centers. Parts must be grouped into the part families to facilitate flow based production. We assume that switching from one family to another requires a major change in set-up of the workstation while parts within a family can be accommodated with minor adjustments that require only a relatively small amount of time that can be added to the processing time, hence the set-up time between the families is sequence-dependent. This assumption reflects the fact that each family includes a set of parts with different demand and due date requirements but share the same set-up, processing and routing requirements. For the rest of the paper, part j of family i is denoted as item (i, j) . All the parts in a family follow the same routing on a modified flowline. In a modified flowline, similar to flowlines, parts follow a uni-directional flow, but contrary to flowlines, they do not have the same routing. A more detailed discussion on the formation of part families and corresponding manufacturing cells can be found in Akturk and Balkose (1996). We number the stages starting from the last stage of production, i.e., stage 1 is the final stage and for any p and m in the routing of item (i, j) if $p < m$, p succeeds m . The notation used throughout the paper is given in Appendix.

3. The proposed algorithm

There are three decision variables: the withdrawal cycle length, T , the number of kanbans for item (i, j) at stage m for a given T , n_{ijm}^T , and the kanban sizes, a_{ij}^T . Note that the kanban sizes are independent of the stage, i.e. for a given withdrawal cycle length T , once set the kanban sizes are kept constant throughout the system. However, the number of kanbans allocated to each stage can be different. The objective is to minimize the total cost that is the sum of the back-order and inventory holding costs over the planning horizon. The proposed algorithm generates several alternatives for each of the decision variables and chooses the best combination among these by comparing the total cost of alternatives.

The main steps of the proposed algorithm are as follows:

- Step 1.* Generate a set of alternatives for withdrawal cycle length. For all possible values of withdrawal cycle length, T ,
- 1.1. Find the number of withdrawal cycles.
 - 1.2. Estimate lead time and calculate the maximum inventory levels.
 - 1.3. Form kanban size alternatives for each item. For each kanban size alternative,
 - 1.3.1. For each withdrawal cycle, call procedure scheduling.
 - 1.3.2. Find the total cost of the schedule found in the previous step.
- Step 2.* Select the alternative with the minimum total cost.

In step 1, we generate a set of alternatives for the withdrawal cycle length, P , as follows:

$$P = \left\{ T : T = \left\lceil \frac{T_{\max}}{A} \right\rceil, \forall A = \{1, 2, 4, 8, \dots, 2^r\} \right\}$$

where $\lceil \cdot \rceil$ gives the smallest integer greater than or equal to the operand and T_{\max} is the length of each period. We assume that the longest withdrawal cycle length is a period.

In step 1.1, given a withdrawal cycle length T , the number of withdrawal cycles for each period is calculated as $\lceil T_{\max}/T \rceil$. Then, the number of withdrawal cycles over the planning horizon is $(\lceil T_{\max}/T \rceil \times \text{planning horizon})$. In step 1.2, we calculate the maximum inventory level of each item at each stage by using the expected period demand. This step has two levels. In the first level, lead times are estimated in terms of number of periods for each stage. In the second level, by using these estimates and Toyota formula, the maximum inventory levels are determined.

For the lead time estimation, Ragatz and Mabert (1984) show that the rules that utilize shop information generally perform better than rules that utilize only job information. In the literature there are no models that can directly be applied to periodic review systems. Therefore, we select the work-in-queue (WIQ) rule proposed by Ragatz and Mabert and modify it to adapt to periodic review systems. According to the WIQ rule, the lead time of item (i, j) at stage m , F_{ijm} , is estimated as $F_{ijm} = k_m \cdot \text{WIQ}_m$ where WIQ_m is WIQ of stage m , and k_m is the lead time estimation coefficient for stage m . We then convert this lead time estimate to number of periods as follows: $L_m = F_{ijm}/T_{\max}$, if $F_{ijm} > T_{\max}$, and 1, otherwise. As the lead times are estimated for each stage, the maximum inventory levels for an item at each stage may be different. In that way, we can allocate different number of kanbans at each stage for the same item. This will not only increase the solution space for the scheduling but also may increase the power of the kanban system over imperfect settings (Takahashi 1994).

Kanban systems maintain constant levels of total on-hand and in-process inventory at each production stage. To determine the maximum inventory level, we use the Toyota formula:

$$\text{maximum inventory level} = n \cdot a = D \cdot L \cdot (1 + s)$$

where n is the number of kanbans, a is the kanban size, D is the average demand rate, L is the expected lead time, and s is the safety factor. An important problem that arises with the Toyota formula is the estimation of the lead times. Lead time is not an attribute of the part; rather it is a property of the shop floor. Lead times vary greatly

depending on capacity, shop load, product mix and batch sizes (Karmarkar 1993). Furthermore, Karmarkar and Kekre (1989) show that the number of kanbans and kanban sizes have a significant effect on the performance of the kanban system, and it is not possible to think L independent of these two variables. Therefore, the maximum inventory level of each item at its final stage is calculated by using the lead time estimate for its final stage, L_1 , along with the expected period demand and the safety factor s as follows:

$$\text{MAXINV}_{ij1} = \lceil \text{expected period demand} * L_1 * (1 + s) \rceil.$$

The maximum inventory levels at upstream stages will be found with backward propagation as follows: $\text{MAXINV}_{ijm} = \lceil L_m * \text{MAXINV}_{ijp} \rceil$ where p is the succeeding stage in the routing of item (i, j) , i.e. $p < m$, and the production stages are numbered backwards. At any withdrawal cycle, the maximum amount that stage p demands from stage m is MAXINV_{ijp} . As the information lead time of the kanban systems are long, the upstream stages cannot react to changes in demand easily and the system becomes erratic. Huang *et al.* (1983) show that the demand variability is amplified in a multi-stage setting. In order to stabilize this erratic behavior, instead of using the mean demand to set the inventory levels at each stage, we use the maximum demand that may occur at any given time. Hence, the algorithm may allocate more inventories to upstream stages.

In step 1.3, we generate the alternatives for the number of kanbans and kanban sizes in such a way that for all alternatives $n_{ijm} a_{ij}^T \sim \text{MAXINV}_{ijm}$. The relationship is not a strict equality due to rounding errors. This allows a fair comparison among the alternatives generated at this step. Step 1.3, generates the set of kanban size alternatives, A_{ij}^T , for each item as follows:

$$A_{ij}^T = \left\{ a_{ij}^T : a_{ij}^T = \left\lceil \frac{\text{MAXINV}_{ij1}}{n_{ij1}^T} \right\rceil, \forall n_{ij1}^T = \{1, 2, 4, 8 \dots, 2^r\} \right\}.$$

Step 1.3.1 finds the schedules for each withdrawal cycle. The flowchart of the scheduling module, which is repeated for all stages, is given in figure 1. Before the execution of the main steps of scheduling, some of the sets should be updated. The demand for the withdrawal cycle is calculated as

$$D_{ij1}^t = \left\lceil \frac{d_{ij}^t - \text{FG}_{ij}^{t-1}}{a_{ij}^T} \right\rceil$$

where $\text{FG}_{ij}^t = D_{ij1}^t \cdot a_{ij}^T - (d_{ij}^t - \text{FG}_{ij}^{t-1})$.

There are three main levels in the scheduling module. At level 1, we try to schedule all the items waiting to be processed in front of stage m in withdrawal cycle t . We call this set the schedule set, SS_m^t . To do this, we first update the inventory levels and back-orders as follows: $I_{ijm}^t = \max(0, S_{ijm}^{t-1} - D_{ijm}^t - B_{ijm}^{t-1})$ and $B_{ijm}^t = \max(0, D_{ijm}^t + B_{ijm}^{t-1} - S_{ijm}^{t-1})$ where $S_{ijm}^{t-1} = I_{ijm}^{t-1} + \text{SCH}_{ijm}^{t-1}$. The back-order cost and inventory holding cost at each stage are calculated as follows: back-order cost at stage m is $\sum_{i=1}^I \sum_{j=1}^{\text{size}[i]} B_{ijm}^t \cdot a_{ij}^T \cdot b_{ijm}$ and inventory holding cost at stage m is $\sum_{i=1}^I \sum_{j=1}^{\text{size}[i]} I_{ijm}^t \cdot a_{ij}^T \cdot h_{ijm}$. The number of kanbans for item (i, j) waiting to be processed in front of stage m in withdrawal cycle t is updated as $\text{UNS}_{ijm}^t = \text{UNS}_{ijm}^{t-1} + D_{ijm}^t$.

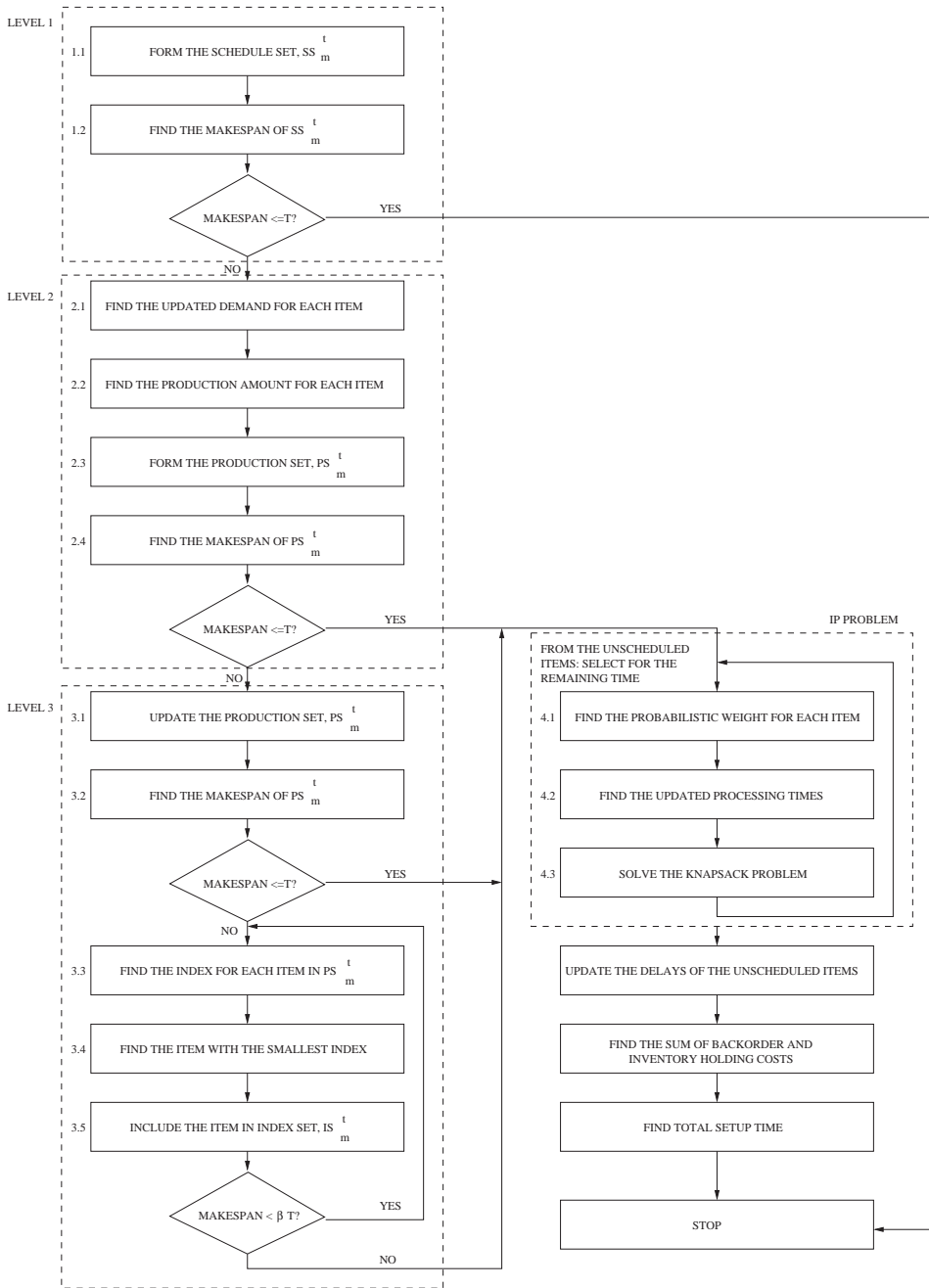


Figure 1. Flowchart of the scheduling module.

Let us describe these three levels briefly for stage m :

Level 1: Generate the schedule set, SS_m^t . This set consists of all the items waiting to be processed in front of stage m . If all the items in SS_m^t cannot be processed within the given withdrawal cycle length, T , i.e. the makespan of the schedule set SS_m^t , denoted as $MK(SS_m^t)$, is greater than T , then goto Level 2.

Level 2: Narrow down the schedule set to a new set called the production set, PS'_m . The production set includes the items that should be scheduled to prevent future back-orders. If all the items in PS'_m cannot be processed within the given withdrawal cycle length, T , then go to level 3.

Level 3: Narrow down the production set to a new set called the index set, IS'_m . The index set schedules the items in order to minimize the total back-order costs.

The details of these levels are given in figure 1 and are also discussed below. In level 1, the makespan of a schedule set is found by using the nearest neighbour (NN) rule to minimize the sequence-dependent set-up times, such that we group all the items in the schedule set to their associated families, i.e. intrafamily splits are not allowed, and find a sequence of families by always selecting the unscheduled family with the smallest set-up time.

At level 2, we narrow down the schedule set to a new set called production set, PS'_m . The production set includes the items that should be scheduled to prevent future back-orders. Since the whole schedule set, SS'_m , cannot be completed on time, back-orders may occur. Stages will become interdependent due to back-orders and to deal with this interdependency, a production amount that considers the expected demand in the future is used. In accordance with the kanban pull logic, we only consider the actual demand information in this withdrawal cycle while solving the kanban scheduling problem for stage m . In order to lower the probability of future back-orders and incorporate a global perspective into the schedule, we estimate the expected demand in the future by utilizing the past withdrawal cycles' demand information so that we can reflect the demand trend. We calculate updated demands for each item as follows:

$$d_{ijm}^u = \gamma \cdot \frac{\sum_{k=1}^c D_{ijm}^{t-k}}{c} + (1 - \gamma) \cdot D_{ijm}^t$$

where γ is a constant such that $0 < \gamma < 1$ and c is the number of past withdrawal cycles used in forecasting. Consequently we find the minimum amount of each item, PA_{ijm} , that should be scheduled to avoid future back-orders by using the updated demands, i.e. $PA_{ijm} = d_{ijm}^u - S_{ijm}^{t-1} + B_{ijm}^{t-1}$. We include all the items with positive production amounts to the production set, which automatically includes all the items that are currently under back-order. We try to schedule all the items in the production set. If it is not possible to schedule all the items, we proceed to level 3. Otherwise, we solve a set of integer programming (IP) problems to select a subset of items from the schedule set that are not included to the production set, $SS'_m - PS'_m$, to fill the remaining time.

At level 3, we first update the production set PS'_m . If the production amount of an item is bigger than its expected demand, we set the production amount to the expected demand. We again narrow down the production set to a new set called index set by means of a proposed ranking index. The aim at this level is to minimize the back-order cost. The proposed ranking index, α_{ijm} , is calculated as:

$$\alpha_{ijm} = \frac{\text{set-up time of family } i}{\sum_{j=1}^{\text{size}[i]} \sum_{s=0}^{MS_{ijm}} (s+1) \cdot Q_{ijm}[s] \cdot b_{ijm}} + \frac{P_{ijm}}{(1 + MS_{ijm}) \cdot b_{ijm}}$$

In kanban systems, the in-process inventories should always be full. Even if there is no demand, there may be an order for an item which has lower in-process inventory than the maximum inventory level. For these items, there is no back-order, so their

delays are expressed as zero and $Q_{ijm}[0]$ is the amount needed to fill up the in-process inventories. The proposed index considers a possible trade-off between a set-up time lost for a family with the urgency of an item. In the index, back-order costs are assumed to be the weights for each item. For the families, the total back-order cost is summed over all items. The first term of the index allocates the family set-up to each item in the family, which is the same for all items in a family, whereas the second term is item specific. We select the item with the smallest index and include it to the index set, IS_m^t . We use the index to form the initial set till $\beta\%$ of the withdrawal cycle is occupied. For the remaining time, we first schedule items from the production set, which are not included to the index set, $PS_m^t - IS_m^t$, by solving a set of integer programming problems. If time remains, we schedule items from the schedule set that are not included to the index set, $SS_m^t - IS_m^t$. β is a predefined number between 0 and 1 and determines the number of IP problems that will be solved. Even though the index is dynamic, it is myopic in nature. There is a trade-off between using the index and IP formulation. The index is faster but the IP formulation eliminates the myopicity. When β is small, we solve more IPs which increases the computation times but also improves the solution quality.

After a number of items are scheduled at level 2 or 3, a number of IP problems are used to schedule items for the remaining time. After level 2, we first find a probabilistic weight for each unscheduled item from set $SS_m^t - PS_m^t$ such that

$$w_{ijm} = [\text{Prob}(D_{ijm}^{t+1} > PS_{ijm}^t + I_{ijm}^t) \cdot b_{ijm} - \text{Prob}(D_{ijm}^{t+1} < PS_{ijm}^t + I_{ijm}^t) \cdot h_{ijm}] \cdot a_{ij}^T.$$

For a given probability mass function of the demand distribution, the probabilistic weight is the difference between the probability that an item will be back-ordered in the next withdrawal cycle multiplied with its back-order cost and the probability that it will remain unused multiplied with its inventory holding cost. Then we find the updated processing times for all unscheduled items with positive probabilistic weights such that $p_{ijm}^u = p_{ijm} \cdot a_{ij}^T$. As described earlier, we only incur a sequence-dependent set-up time between the families. Therefore, we calculate the required set-up times of the unscheduled families with respect to the family that is scheduled to the last position in the partial sequence. We solve the following IP formulation to find the next item to be scheduled. In this step, the aim is to select the most profitable item among the unscheduled ones. The probabilistic weights together with the updated processing times and set-up times, if any, are used to decide if it will be profitable to produce the item in the remaining time slot:

$$\begin{aligned} &\text{Maximize } \sum_{i=1}^I \sum_{j=1}^{\text{size}[i]} (X_{ijm} \cdot w_{ijm}) \\ &\text{Subject to } \sum_{i=1}^I (Y_{im} \cdot \text{set-up time of family } i) \\ &\quad + \sum_{i=1}^I \sum_{j=1}^{\text{size}[i]} (X_{ijm} \cdot p_{ijm}^u) \leq (T - \text{MK}(PS_m^t)) \\ &\quad X_{ijm} - \text{RM}_{ijm} \cdot Y_{im} \leq 0 \\ &\quad \sum_{i=1}^I Y_{im} \leq 1 \end{aligned}$$

where X_{ijm} is an integer variable and gives the number of kanbans of item (i, j) at stage m that should be included to PS'_m . In PS'_m , there are two types of items. For the first type, for every unscheduled item there is at least one item from its corresponding family that is already scheduled, hence there is no need to allocate a time for machine set-up. We only define a binary variable Y_{im} for the unscheduled families that require an additional set-up time, so Y_{im} takes value of 1 when an item from family i is scheduled. RM_{ijm} is total number of kanbans of item (i, j) in the set $SS'_m - PS'_m$. As the probabilistic weights might be different for each kanban of an item, we can dynamically assign weights to each kanban. If a new family is scheduled, then we update the IP problem and continue solving until the remaining time is not enough to produce an unscheduled item. Finally, we update the delay values for the unscheduled items, where delay is defined as the number of withdrawal cycles that an item is back-logged.

After level 3, a similar set of IP problems are solved in order to schedule the items from set $PS'_m - IS'_m$. After level 2, the probabilistic weights can be negative or positive, while after level 3, the weights of items in set $PS'_m - IS'_m$ are strictly positive due to updating the production set prior to index calculation. After the items in set $PS'_m - IS'_m$ are considered, if time remains, then the items from set $SS'_m - IS'_m$ are considered. But, the probabilistic weights of these items are calculated as $w_{ijm} = b_{ijm} - h_{ijm}$.

At the end of the three levels, we have scheduled a number of items, denoted as set SCH'_{ijm} , at stage m for withdrawal cycle t . The demand of the preceding stage p , ($p > m$), is the set of scheduled items at stage m . After all stages and the withdrawal cycles are scheduled, total back-order and inventory holding costs are calculated. At the end of the algorithm, we select the alternative giving minimum cost. As discussed earlier, several models in the literature assume station independence through either assuming infinite capacity or not allowing back-orders as summarized in Akturk and Erhun (1999). Under JIT system, all stages are integrally tied to each other and if there is a delay in one stage then both the preceding and succeeding stages should be affected to highlight interdependencies among the stages. This is crucial for reducing the size of the inventory levels and eliminating overall waste. However, this possibility is eliminated with the assumption of no back-orders and an infinite production capacity. Therefore, in the proposed algorithm a scheduling decision in one stage affects both the preceding and succeeding stages to highlight interdependencies among the stages.

4. Numerical Example

In this section, we will explain the execution of the scheduling module over a simple example. It is assumed that the withdrawal cycle length, kanban sizes and the number of kanbans are known. Let the withdrawal cycle length be $T_{\max}/2 = 480/2 = 240$ minutes. The data for the example are given in table 1 for three families, five parts and three withdrawal cycles. There is only one stage therefore we drop the index m for simplicity. In table 2, the sequence-dependent set-up times between the families are given. The first row in the table corresponds to the case where the stage is not initially set-up to any of the families.

The number of kanbans demanded for item $(1, 1)$ in the first withdrawal cycle is $D^1_{11} = \lceil (30 - 0)/10 \rceil = 3$ and $FG^1_{11} = 3 \cdot 10 - 30 = 0$. The rest of the calculations are similar. So the demands for each withdrawal cycle will be $D^1 = \{3, 3; 3, 3, 3\}$ and $FG^1 = \{0, 3; 7, 5, 0\}$, $D^2 = \{1, 1; 3, 3, 2\}$ and $FG^2 = (3, 0; 5; 8, 0)$, and

| Family Part | 1 | | 2 | 3 | |
|--|-------------|-------------|--------------|--------------|--------------|
| | 1 | 2 | 1 | 1 | 2 |
| Kanban size, a_{ij} (units) | 10 | 10 | 10 | 10 | 10 |
| Maximum inventory level (units) | 20 | 20 | 20 | 20 | 20 |
| Back-order cost, b_{ij} (\$/unit/withdrawal cycle) | 1 | 2 | 10 | 2 | 3 |
| Processing times, p_{ij} (minutes/unit) | 1.5 | 1 | 4 | 1 | 2 |
| Demand, d_{ij}^t , ($t = 1, t = 2, t = 3$) | (30, 7, 15) | (27, 13, 8) | (23, 32, 20) | (25, 27, 17) | (30, 20, 11) |

Table 1. Data for numerical example.

| | 1 | 2 | 3 |
|---|----|----|----|
| 0 | 5 | 5 | 5 |
| 1 | 0 | 40 | 15 |
| 2 | 5 | 0 | 5 |
| 3 | 10 | 20 | 0 |

Table 2. The sequence-dependent set-up times (in minutes).

$D^3 = \{2, 1; 2; 1, 2\}$ and $FG^3 = \{8, 2; 5; 1, 9\}$. Initially the set of inventories, back-orders and unscheduled items will be $I^0 = \{2, 2; 2; 2, 2\}$, $B^0 = \{0, 0; 0; 0, 0\}$ and $UNS^0 = \{0, 0; 0; 0, 0\}$, respectively. In the set presentation, (;) is used to separate the families and (,) is used to separate the items within the families.

Withdrawal cycle 1: First, we should update the inventory levels, back-orders and schedule set as discussed earlier:

$$\begin{aligned}
 I_{11}^1 &= \max(0, 2 - 3 - 0) = 0, \quad B_{11}^1 = \max(0, 3 + 0 - 2) = 1, \\
 SS_{11}^1 &= UNS_{11}^0 + D_{11}^1 = 0 + 3 = 3, \quad I_{12}^1 = I_{21}^1 = I_{31}^1 = I_{32}^1 = 0, \\
 B_{12}^1 &= B_{21}^1 = B_{31}^1 = B_{32}^1 = 1,
 \end{aligned}$$

and $SS_{12}^1 = SS_{21}^1 = SS_{31}^1 = SS_{32}^1 = 3$. Then, the back-order cost will be equal to 180 (back-order cost = $1 \cdot 10 \cdot 1 + 1 \cdot 10 \cdot 2 + 1 \cdot 10 \cdot 10 + 1 \cdot 10 \cdot 2 + 1 \cdot 10 \cdot 3 = 180$).

We should also calculate the delays of the items. Let $Q[s]$ be the set of $Q_{ij}[s]$. Then, $Q[0] = \{2, 2; 2; 2, 2\}$ and $Q[1] = \{1, 1; 1; 1, 1\}$. Due to space limitations, the above calculations will not be shown explicitly in the rest of the example, although the calculations are very similar.

Level 1: Find the makespan for the complete schedule set SS^1 . The sequence according to the NN rule will be family 2, family 3, and family 1, and the makespan is $\mathbf{5} + 4 \cdot 3 \cdot 10 + \mathbf{5} + 1 \cdot 3 \cdot 10 + 2 \cdot 3 \cdot 10 + \mathbf{10} + 1.5 \cdot 3 \cdot 10 + 1 \cdot 3 \cdot 10 = 305$ minutes, where the bold numbers correspond to set-up times and the products correspond to the total production time for the kanbans demanded. As the makespan of SS^1 is longer than T which is 240 minutes, it is not possible to schedule all the items in this set, so proceed to level 2.

Level 2: Determine the new set, PS^1 . As we have only one withdrawal cycle, the updated demands of the items will be equal to the withdrawal cycle demands, which are 3. As the inventories are empty, the production amounts will be equal to the

updated demands. Therefore, $PS^1 = \{3, 3; 3, 3\}$ and it will not be possible to schedule all of the items, so proceed to level 3.

Level 3: Calculate the proposed ranking index for each item and form the new schedule set IS^1 . The total back-order costs of families 1, 2 and 3 are equal to 120, 400 and 200, respectively. The ranking indexes will be: $\alpha_{11} = \frac{5}{120} + \frac{1.5}{2} = 0.79$, $\alpha_{12} = \frac{5}{120} + \frac{1}{4} = 0.29$, $\alpha_{21} = \frac{5}{400} + \frac{4}{20} = \mathbf{0.21}$, $\alpha_{31} = \frac{5}{200} + \frac{1}{4} = 0.28$, $\alpha_{32} = \frac{5}{200} + \frac{2}{6} = 0.36$. In the index calculations, the bold number shows the minimum ranking index, hence the item that should be selected. Therefore, item (2, 1) is scheduled first and its completion time will be 45 minutes. The total back-order cost of the family 2 will decrease to 200 and the other costs will remain unchanged. We schedule one kanban at a time. The reason for this is to assign dynamic back-order costs to items. There can be kanbans with different delay values for the same item, so these kanbans will have different back-order costs. For example, the back-order cost of item (2, 1) was 20 in the first index calculation. When we recalculate the index, as we have already scheduled the kanban of item (2, 1) that is one withdrawal cycle late, we do not have any back-orders for this item. If we go on scheduling item (2, 1), this will be to avoid the future back-orders that may occur due to empty in-process inventories. So the item (2, 1) will have less priority in the next calculations of the index. The new indexes will be $\alpha = \{0.79, 0.29; 0.40; \mathbf{0.28}, 0.36\}$. The completion time of item (3, 1) will be 60 minutes. The total back-order cost for family 3 will be 160. The rest of the calculations are given below:

| Step | α | Selected item | Completion time |
|------|--|---------------|-----------------|
| 3 | {0.83, 0.33; 0.40; 0.50, 0.33 } | (3, 2) | 80 |
| 4 | {0.83, 0.33 ; 0.40; 0.50, 0.66} | (1, 2) | 100 |
| 5 | {0.75, 0.50; 0.40 ; 0.50, 0.66} | (2, 1) | 140 |
| 6 | {0.75, 0.50; 0.40 ; 0.50, 0.66} | (2, 1) | 180 |
| 7 | {0.75, 0.50 ; -; 0.50, 0.66} | (1, 2) | 190 |
| 8 | {0.75, 0.50 ; -; 0.50, 0.66} | (1, 2) | 200 |
| 9 | {0.75, -; -; 0.50 , 0.66} | (3, 1) | 210 |
| 10 | {0.75, -; -; 0.50 , 0.66} | (3, 1) | 220 |
| 11 | {0.75, -; -; -; 0.66 } | (3, 2) | 240 |

We reach to the end of time period, so we stop after Step 11. The index set will be $IS^1 = \{0, 3; 3, 2\}$. The sequence will be family 2, family 3 and family 1, which is the same as the sequence found by the NN rule. Therefore, the makespan will be 240 minutes. The inventories and unscheduled items at the end of first withdrawal cycle will be $S^1 = \{0, 3; 3, 2\}$ and $UNS^1 = \{3, 0; 0, 0, 1\}$, respectively. The stage is currently set-up to family 1 and the back-order cost for the first withdrawal cycle is 180.

Withdrawal cycle 2: For the second withdrawal cycle, the updated inventory levels, back-orders, and schedule set will be $I^2 = \{0, 1; 0; 0, 0\}$, $B^2 = \{2, 0; 1; 1, 1\}$ and $SS^2 = \{\mathbf{3} + 1, 1; 3; 3, \mathbf{1} + 2\}$, respectively. The back-order cost for the second withdrawal cycle is 170. The delays will be $Q[0] = \{2, 1; 2; 2, 2\}$, $Q[1] = \{1, 0; 1; 1, 1\}$ and $Q[2] = \{1, 0; 0; 0, 0\}$.

Level 1: The sequence of SS^2 according to the NN rule will be family 1, family 3, and family 2, and the makespan will be $315 > T$, so proceed to level 2.

Level 2: The updated demand for item (1, 1) will be $d_{11}^u = \lceil 0.5 \cdot 3 + 0.5 \cdot 1 \rceil = 2$. The rest of the calculations will be similar. So, the updated demands will be

$d'' = \{2, 2; 3; 3, 3\}$. The production amounts are equal to the difference between updated demand and in-process inventory when positive, and zero otherwise. Hence $PS^2 = (d'' - I^2)^+ = \{2, 1; 3; 3, 3\}$. The makespan of this set will be $285 > T$, so proceed to level 3.

Level 3: The index calculations will not be shown explicitly. The index set will be $IS^2 = \{1, 1; 3; 3, 1\}$, and the makespan will be 230. To fill the remaining 10 minutes, an IP problem will be solved. We first calculate the updated processing times. As at least one item is scheduled from each family, the updated processing times will be equal to the following processing times: $p'' = \{15, 10; 40; 10, 20\}$. The set of unscheduled items, RM, will be $PS' - IS' = \{1, 0; 0; 0, 2\}$. As the remaining time is 10 minutes and $RM = \{1, 0; 0; 0, 2\}$, no other item can be added to the index set from the set $PS' - IS'$. We now consider the set $SS' - IS'$, and the set of unscheduled items will be $\{3, 0; 0; 0, 2\}$. Given the remaining time and RM, no items can be included to the index set from the set $SS' - IS'$. Hence, the final index set will be $IS^2 = \{1, 1; 3; 3, 1\}$. The inventories and unscheduled items will be $S^2 = \{1, 2; 3; 3, 1\}$ and $UNS^2 = \{3, 0; 0; 0, 2\}$, respectively. The total back-order cost for two withdrawal cycles will be 350.

In order to demonstrate how the β parameter might affect the final schedule as well as its total cost, we also schedule the second withdrawal cycle with $\beta = 0.5$. We will use the proposed ranking index to form the index set, IS^2 , till the total processing time is less than or equal to $\beta \times 240 = 120$. So, the previous index calculations will be valid till total processing time reaches 120. Therefore, we will begin to solve IP problems after the third step. At that time the index set was $IS^2 = \{0, 0; 1; 1, 1\}$, after applying the NN rule the order of items was $\{\text{item}(3, 1) - \text{item}(3, 2) - \text{item}(2, 1)\}$ and the makespan = $15 + 10 + 20 + 20 + 40 = 105$ minutes. The updated processing times for the set of remaining items, $RM = \{2, 1; 2; 2, 2\}$, will be $p'' = \{15, 10; 40; 10, 20\}$. Since there is no item scheduled from the first family, the sequence dependent set-up time of this family, 5 minutes, should be considered in the IP formulation. In order to calculate the probabilistic weights, let's assume that probability mass function of the demand distribution is given as $f_D(d) = \{0.3 \text{ if } D_{ij}^t = 1, 0.4 \text{ if } D_{ij}^t = 2, \text{ and } 0.3 \text{ if } D_{ij}^t = 3\}$. The probabilistic weights will be $w_{ij} = \text{Prob}(D_{ij}^{t+1} \geq IS_{ij}^t + I_{ij}^t) \cdot b_{ij} \cdot a_{ij}$, or $w = \{1 \times 10 = 10, 0.7 \times 20 = 14; 0.7 \times 100 = 70; 0.7 \times 20 = 14, 0.7 \times 30 = 21\}$.

Therefore we solve the following IP model:

$$\begin{aligned} \text{Maximize} \quad & 10X_{11} + 14X_{12} + 70X_{21} + 14X_{31} + 21X_{32} \\ \text{Subject to} \quad & 15X_{11} + 10X_{12} + 40X_{21} + 10X_{31} + 20X_{32} + 5Y_1 \leq 135 \\ & 0 \leq X_{21} \leq 2, 0 \leq X_{31} \leq 2, 0 \leq X_{32} \leq 2 \\ & 0 \leq X_{11} \leq 2 \cdot Y_1, 0 \leq X_{12} \leq 1 \cdot Y_1, Y_1 \leq 1 \end{aligned}$$

The solution is $X_{11} = 0, X_{21} = X_{31} = 2, X_{12} = X_{32} = 1$ and $Y_1 = 1$. The index set will become $IS^2 = \{0, 1; 3; 3, 2\}$ with the makespan of 235. The remaining time, 5 minutes, is not enough to produce any unscheduled item. Therefore, we will not solve an IP problem for the set $SS^2 - IS^2$. When $\beta = 0.5$, the inventories and unscheduled items will be $S^2 = \{0, 2; 3; 3, 2\}$ and $UNS^2 = \{4, 0; 0; 0, 1\}$, respectively. The total back-order cost for two withdrawal cycles will be 350.

Withdrawal cycle 3: The updated inventory levels, back-orders and schedule set for the third withdrawal cycle when $\beta = 1$ will be $I^3 = \{0, 1; 0; 1, 0\}$,

| | t | I^t | B^t | SS^t | S^t | UNS^t | cum. back. cost |
|----------|-----|-----------------|-----------------|---------------------------------|-----------------|-----------------|-----------------------|
| SPT/LATE | 1 | {0, 0; 0, 0, 0} | {1, 1; 1, 1, 1} | {3, 3; 3, 3, 3} | {3, 3; 0; 3, 3} | {0, 0; 3; 0, 0} | 180 |
| | 2 | {1, 1; 0; 0, 0} | {0, 0; 4; 1, 0} | {1, 1; 3 + 3; 3, 2} | {2, 2; 2; 3, 2} | {0, 0; 4; 0, 0} | 600 |
| | 3 | {0, 1; 0; 1, 0} | {0, 0; 4; 0, 0} | {2, 1; 4 + 2; 1, 2} | {2, 2; 2; 2, 2} | {0, 0; 4; 0, 0} | 1000 |
| FCFS | 1 | {0, 0; 0; 0, 0} | {1, 1; 1, 1, 1} | {3, 3; 3; 3, 3} | {3, 3; 1; 3, 1} | {0, 0; 2; 0, 2} | 180 |
| | 2 | {1, 1; 0; 0, 0} | {0, 0; 3; 1, 2} | {1, 1; 2 + 3; 3, 2 + 2} | {2, 2; 3; 1, 2} | {0, 0; 2; 2, 2} | 560 |
| | 3 | {0, 1; 0; 0, 0} | {0, 0; 2; 1, 2} | {2, 1; 2 + 2; 2 + 1, 2 + 2} | {2, 2; 2; 2, 2} | {0, 0; 2; 1, 2} | 840 |
| SPT-F | 1 | {0, 0; 0; 0, 0} | {1, 1; 1, 1, 1} | {3, 3; 3; 3, 3} | {3, 3; 0; 3, 3} | {0, 0; 3; 0, 0} | 180 |
| | 2 | {1, 1; 0; 0, 0} | {0, 0; 4; 1, 0} | {1, 1; 3 + 3; 3, 2} | {2, 2; 2; 3, 2} | {0, 0; 4; 0, 0} | 600 |
| | 3 | {0, 1; 0; 1, 0} | {0, 0; 4; 0, 0} | {2, 1; 4 + 2; 1, 2} | {2, 2; 2; 2, 2} | {0, 0; 4; 0, 0} | 1000 |
| FCFS-F | 1 | {0, 0; 0; 0, 0} | {1, 1; 1, 1, 1} | {3, 3; 3; 3, 3} | {3, 3; 0; 3, 3} | {0, 0; 3; 0, 0} | 180 |
| | 2 | {1, 1; 0; 0, 0} | {0, 0; 4; 1, 0} | {1, 1; 3 + 3; 3, 2} | {1, 1; 6; 0, 0} | {1, 1; 0; 3, 2} | 600 |
| | 3 | {0, 0; 0; 0, 0} | {1, 0; 0; 2, 2} | {1 + 2, 1 + 1; 2; 3 + 1, 2 + 2} | {3, 2; 0; 4, 4} | {0, 0; 2; 0, 0} | 710 |

Table 3. The execution of algorithms.

$B^3 = \{3, 0; 0; 0, 2\}$, and $SS^3 = \{3 + 2, 1; 2; 1, 2 + 2\}$, respectively. The back-order cost for this withdrawal cycle will be 90. The total back-order cost for three withdrawal cycles is 440.

For $\beta = 0.5$, the updated the inventory levels, back-orders, and schedule set of third withdrawal cycle will be $I^3 = \{0, 1; 0; 1, 0\}$, $B^3 = \{4, 0; 0; 0, 1\}$, and $SS^3 = \{4 + 2, 1; 2; 1, 1 + 2\}$. The back-order cost will be 70. The total back-order cost for three withdrawal cycles will be 420.

There are no algorithms in the existing literature that we can directly compare our results with. Therefore, we will compare our study with the four widely used sequencing rules in the literature, which are SPT/LATE, SPT-F, FCFS, and FCFS-F. For a detailed discussion of why we select these rules and an algorithmic description of each rule, refer to Akturk and Erhun (1999). The execution of the algorithms for the numerical example can be seen in table 3.

In summary, the total back-order costs over three withdrawal cycles are 840, 1000, 710, and 1000 for FCFS, SPT/LATE, FCFS-F, SPT-F, respectively. For the proposed algorithm, when $\beta = 1$, the back-order cost is 440, and when $\beta = 0.5$ in the second withdrawal cycle, the back-order cost is 420. This simple example highlights the effectiveness of the proposed algorithm and the ranking index. As expected, when β decreases, the total back-order costs also decrease.

5. Computational analysis

An experimental design is developed to test the efficiency of the proposed algorithm by comparing with the FCFS, FCFS-F, SPT/LATE, and SPT-F rules. All of the algorithms are coded in C language and compiled with Gnu C compiler. The IP formulations in the proposed algorithm are solved by using callable library routines of CPLEX 5.0 MIP solver on a Sun Ultra 4000 workstation. In this section, first the experimental factors are given. Then, the parameters and the way how they are determined are explained. Finally, the computational results are provided.

5.1. Experimental factors

In the experimental design, seven factors that can affect the efficiency of kanban systems are considered. The first factor is the number of families. As the number of

families increase, the set-up requirement increases and the scheduling decision becomes more important. The second and third factors are the demand mean and variability, respectively. The probability mass function (pmf) of the demand distribution for low and high variability cases are given as:

$$f_D^{\text{low}}(d) = \begin{cases} 0.1, & D = \text{UN} \sim [\mu - 13, \mu - 9], \\ 0.2, & D = \text{UN} \sim [\mu - 8, \mu - 4], \\ 0.3, & D = \text{UN} \sim [\mu - 3, \mu + 1], \\ 0.2, & D = \text{UN} \sim [\mu + 2, \mu + 6], \\ 0.2, & D = \text{UN} \sim [\mu + 7, \mu + 11], \end{cases}$$

$$f_D^{\text{high}}(d) = \begin{cases} 0.1, & D = \text{UN} \sim [\mu - 26, \mu - 18], \\ 0.2, & D = \text{UN} \sim [\mu - 17, \mu - 9], \\ 0.3, & D = \text{UN} \sim [\mu - 8, \mu + 4], \\ 0.2, & D = \text{UN} \sim [\mu + 5, \mu + 13], \\ 0.2, & D = \text{UN} \sim [\mu + 14, \mu + 22] \end{cases}$$

where μ is the demand mean and $\text{UN} \sim [a, b]$ is uniform distribution in interval $[a, b]$. This density states that for the low-variability case 10% of the time demand will be uniformly distributed in interval $[\mu - 13, \mu - 9]$, and for another 20% of the time it will be uniformly distributed in interval $[\mu - 8, \mu - 4]$, and so on. The demand average for a period is 30 and 50 for the low and high factor levels, respectively. Since a period is equal to the longest withdrawal cycle length, denoted as T_{\max} (480 minutes in our computational setting), the demand values for different withdrawal cycle length alternatives can be found accordingly. The fourth factor is the number of parts in each family. This factor, together with factor F1, affects the product mix and congestion of the shop floor.

The fifth factor specifies the relative load of each stage. In the balanced case, the processing times of items have the same uniform distribution at each stage. In the unbalanced case, the processing times at third stage, stage C, has a uniform distribution with a higher mean. When the processing times are higher, third stage will become a bottleneck stage, hence smooth material flow will be disturbed. The sixth factor is used to determine the sequence-dependent set-up times at each stage. The distribution of set-up time is $\text{UN} \sim [\text{SL}_m, \text{SH}_m]$ where SL_m and SH_m are calculated as follows:

$$\text{SL}_m = \text{S/P} \cdot (\text{average processing time of family } i \text{ at stage } m) \cdot 0.50$$

$$\text{SH}_m = \text{S/P} \cdot (\text{average processing time of family } i \text{ at stage } m) \cdot 1.50$$

where the average processing time of family i at stage m is equal to $(\sum_{j=1}^{\text{size}[i]} p_{ijm} \cdot K) / \text{size}[i]$. K is an estimated kanban size and is taken as 25 or 50 when factor F4 is at its low or high level, respectively, in order to keep the ratio of set-up time to total time constant. The seventh factor is the back-order cost to inventory holding cost (B/I) ratio. The back-order cost of an item at the last stage is equal to the inventory holding cost times the B/I ratio such that $b_{ijm_i} = \text{B/I} \cdot h_{ijm_i}$, where m_i is the index for the last stage of the corresponding item. Back-order and

| Factors | Definition | Low | High |
|---------|--------------------------------|-------------|--------------|
| F1 | Number of families | 4 | 7 |
| F2 | Demand average | 30 | 50 |
| F3 | Demand variability | low | high |
| F4 | Number of parts in each family | UN ~ [4, 8] | UN ~ [8, 12] |
| F5 | Imbalance | Balanced | Unbalanced |
| F6 | S/P ratio | 0.8 | 1.6 |
| F7 | B/I ratio | 10 | 20 |

Table 4. Experimental factors.

inventory holding costs at the last stage are used to calculate the back-order and inventory holding costs at the preceding stages.

The experimental design is a 2^7 full-factorial design. The factors and the values at each level are listed in table 4. Five replications are taken for each combination resulting in 640 different randomly generated runs.

5.2. Fixed parameters

There are five stages, denoted as A, B, C, D and E. The routings for families are fixed and given in figure 2. When factor F1 is at the low level, the first four families are used.

The processing times for balanced case are selected randomly from the interval $UN \sim [0.1, 0.3]$. For unbalanced case, the processing times at stage C are selected randomly from the interval $UN \sim [0.4, 0.8]$ when the number of parts in each family are low, and from the interval $UN \sim [0.3, 0.7]$ when it is high. The forecasting weight is 0.5 and three past withdrawal cycles are used for estimation. The lead time coefficient is same for all stages and equal to 1.01. β is taken to be 1. The planning horizon is 20 days.

The inventory holding costs (\$/unit/period) for the last stage are generated randomly from the interval $UN \sim [6, 12]$. The inventory holding costs in the preceding stages are found by using the inventory holding cost in the last stage such that $[h_{ijm_{i-1}} \ h_{ijm_{i-2}} \ h_{ijm_{i-3}} \ h_{ijm_{i-4}}] = [0.9 \ 0.8 \ 0.7 \ 0.6] \cdot h_{ijm_i}$. For example, the inventory holding cost in third stage, counted from the last stage, $h_{ijm_{i-2}}$, is 0.8 times the inventory holding cost in the last stage, h_{ijm_i} . The inventory holding cost increases as the item goes through the last stage because of the value added to the item. The back-order cost in the preceding stages are such that $[b_{ijm_{i-1}} \ b_{ijm_{i-2}} \ b_{ijm_{i-3}} \ b_{ijm_{i-4}}] = [0.9 \ 0.7 \ 0.5 \ 0.3] \cdot b_{ijm_i}$. We consider back-order costs in the preceding stages, since the back-orders in earlier stages affect the back-order at the last stage and hence the

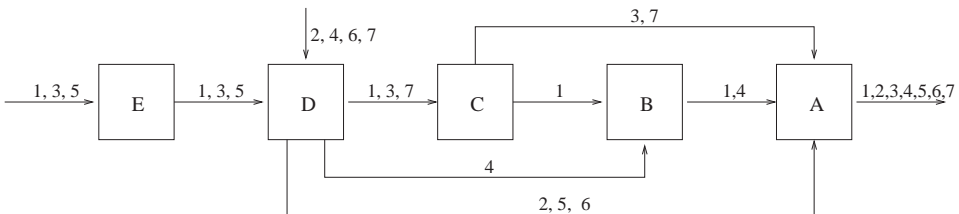


Figure 2. Layout.

total back-order cost in the objective function. The objective is minimizing the total cost which is comprised of back-order and inventory holding costs. The inventory holding cost is the sum of inventory holding costs over all stages for all periods. The back-order cost is found by using the back-orders at the last stage. The effect of the back-orders in earlier stages is considered by multiplying the back-order cost in the last stage by the number of operations in the operation sequence of the item. Since a period is equal to the longest withdrawal cycle length (480 minutes in our computational setting), the inventory holding and back-order costs for different withdrawal cycle length alternatives in minutes can be found accordingly.

The last parameter is the safety factor, s , in Toyota formula which is used to protect against shortages. It is a common conjecture that back-order costs can be decreased by increasing the maximum inventory level or the safety factor in the Toyota formula. This conjecture is valid only for FCFS rule, since it maintains consistent kanban priorities from stage to stage. For other scheduling rules, the best inventory level minimizing the back-order cost, and hence maximizing the customer service level, may not correspond to the highest possible inventory level. For example, when we set s to an arbitrarily large value, a myopic rule like SPT/LATE will select the items with the shortest processing time, and the items with large processing times will be late. The items finished earlier will increase the safety stock limit, and hence have a small effect on the fill rate or back-order. In brief, we can say that in some cases when s increases, the back-order cost may not decrease. The relationship between back-order and inventory holding costs and the safety factor for SPT/LATE and FCFS rules can be seen in figure 3a and 3b, respectively. In this figure, factor combination (1101010) is used as an example,

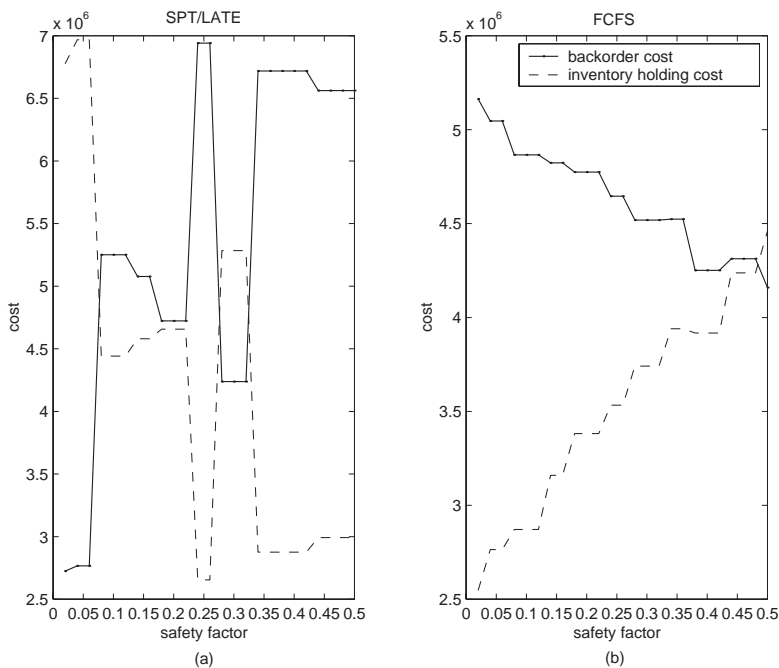


Figure 3. Back-order and inventory holding costs for different dispatching rules.

where 0 and 1 correspond to the low and high levels of each factor, respectively. For the FCFS rule, the back-order cost decreases and inventory holding cost increases as s increases. It is important to note that for the SPT/LATE rule, the back-order cost is not a decreasing function of s . It attains its minimum at $s = 0.02$ and has more than one local minimum. The inventory holding cost does not increase as s increases. It is decreased by decreasing the withdrawal cycle length for higher values of s which can be seen in figure 4a. Since setting s to an arbitrarily large value may not decrease the back-order cost, s should be selected carefully.

In the literature, there is only one study by Co and Sharafali (1997) for the safety factor calculation. They state that the safety stock parameter is constant if the sequence of demands are independent and identically distributed, and give formulas for calculating the safety factor when demand distribution is uniform, exponential, triangular or truncated normal. When demand is uniformly distributed over $[x, y]$ as in our case, the ‘optimal’ s value is calculated as $s = [(b - h) \cdot (y - x)] / [(b + h) \cdot (y + x)]$, where b is the back-order cost and h is the inventory holding cost. For example, the ‘optimal’ s value is equal to 0.033 for the factor combination (1101110), i.e. the demand mean is equal to 50, the demand variability is low, and the B/I ratio is equal to 10 ($b = 10 \cdot h$). When we put b, h, x and y values in the above formula and take the weighted average of five different demand distribution scenarios as it is shown below, then

$$s = \frac{(10h - h)(41 - 37)}{(10h + h)(41 + 37)} 0.1 + \frac{9(46 - 42)}{11(46 + 42)} 0.2 + \frac{9(51 - 47)}{11(51 + 47)} 0.3 + \frac{9(56 - 52)}{11(56 + 52)} 0.2 + \frac{9(61 - 57)}{11(61 + 57)} 0.2$$

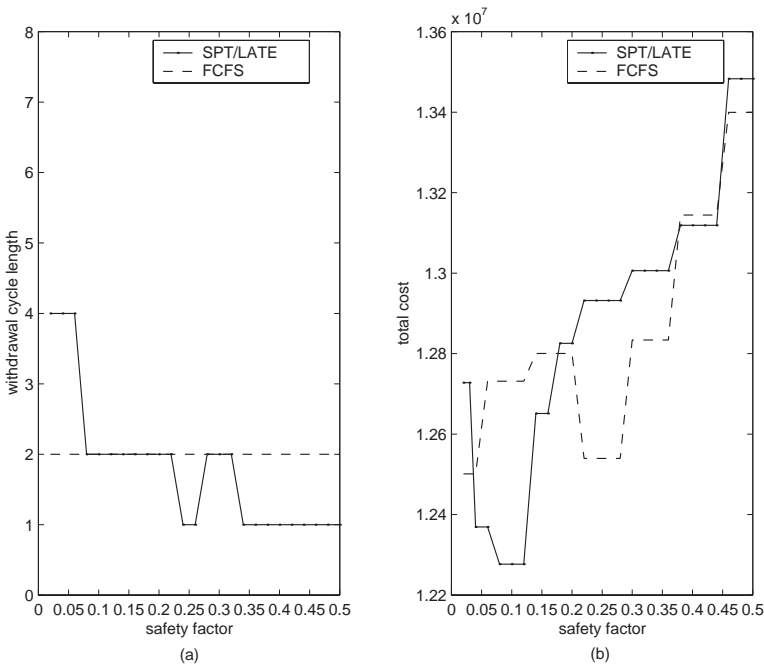


Figure 4. Effect of safety factor on withdrawal cycle length and total cost.

In figure 4b, total cost versus safety factor plot can be seen for the SPT/LATE and FCFS rules for the factor combination (1101110). For the FCFS rule, the minimum total cost is achieved at $s = 0.033$, which is the ‘optimal’ s value according to the above formula. At this safety factor level, FCFS seems better than SPT/LATE in terms of total cost. But, SPT/LATE gives a smaller total cost than FCFS at higher s values (0.08, 0.10, 0.12). Although we showed that the s values calculated according to the formula by Co and Sharafali (1997) are not robust to the selected dispatching rule, we still calculate the s values according to their formula since this is the only study in the literature for the safety factor calculation. The s values change according to the levels of factors F2, F3 and F7, and are calculated as 0.057, 0.063, 0.153, 0.169, 0.033, 0.037, 0.078, 0.086, when the factor combinations for (F2, F3, F7) are (000), (001), (010), (011), (100), (101), (110), (111), respectively.

5.3. Decision variables

There are three decision variables: the withdrawal cycle length, kanban size and number of kanbans. The withdrawal cycle lengths have a significant effect on the average in-process inventories, back-orders and production rates. When kanban sizes are constant, the total cost curve is convex over withdrawal cycle lengths. As withdrawal cycle lengths decrease, the in-process inventories decrease and back-orders increase. This can be seen in figure 5 for the SPT/LATE and FCFS rules for an example run of factor combination (1101111). These algorithms give the minimum cost at different T values which shows the sensitivity of the scheduling policies to withdrawal cycle lengths. Therefore, in this experimental design six alternatives are generated for the withdrawal cycle lengths such as {8, 4, 2, 1, 0.5, 0.25} hours or {480, 240, 120, 60, 30, 15} minutes. There are also six alternatives for the number of kanbans, which are {1, 2, 4, 8, 16, 32}. Thirty-six alternatives for each replication will be evaluated by each sequencing rule in order to find the alternative with the minimum total cost. In table 5, the number of instances of best T values for all runs are summarized. When we analyse the results, we can see that the scheduling policies are not robust to withdrawal cycle lengths. The family based rules SPT-F and FCFS-F prefer relatively shorter withdrawal cycle lengths when they are com-

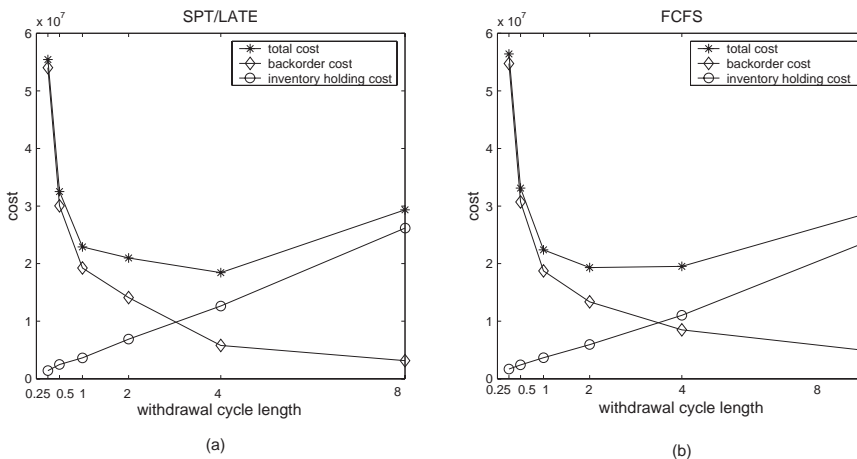


Figure 5. Cost components for a fixed kanban size.

| WCL (hrs) | SPT/LATE | SPTF | FCFS | FCFSF | Algorithm |
|-----------|----------|------|------|-------|-----------|
| 8 | 0 | 0 | 0 | 0 | 0 |
| 4 | 17 | 4 | 15 | 5 | 34 |
| 2 | 62 | 59 | 178 | 114 | 148 |
| 1 | 244 | 236 | 213 | 252 | 214 |
| 0.5 | 254 | 262 | 207 | 242 | 189 |
| 0.25 | 63 | 79 | 27 | 27 | 55 |

Table 5. Number of instances of withdrawal cycle lengths.

pared with SPT/LATE and FCFS. The item-based rules prefer longer withdrawal cycle lengths so that set-up times can be justified.

5.4. Results

We use four measures to compare five algorithms. The first one is (0–1) scaling, where 0 indicates the best value and 1 indicates the worst value among the alternatives for the objective of minimizing the total cost. For each algorithm it is calculated as $S_i = (TC_i - TC_{\min}) / (TC_{\max} - TC_{\min})$, where TC_i is the total cost of algorithm i , TC_{\min} and TC_{\max} are the minimum and maximum of total cost values for each run among all algorithms, respectively. The minimum, average and maximum scaled values for each algorithm can be seen in table 6. According to the average values in the table, the proposed algorithm gives the minimum scale, followed by the FCFS rule. According to the minimum and maximum scales, each algorithm is selected as either the best or the worst algorithm, respectively, in at least one run among 640 runs.

The second measure is eigenvalue normalization. This measure is less sensitive to both the range for the actual values and to the number of data points than the (0–1) scaling. The normalized value, N_i , for each algorithm’s objective value is $N_i = TC_i / (\sqrt{\sum_i TC_i^2})$. In table 6, the minimum, average and maximum normalized values for each algorithm can be seen. When the production environment is suitable for the efficient operation of an algorithm, the normalized values are low. Under the most suitable conditions, FCFS can give the least normalized value. When the factor combinations determine an imperfect environment for the algorithm, the normalized values increase. Therefore, the minimum and maximum normalized values for

| | | SPT/LATE | SPTF | FCFS | FCFSF | Algorithm |
|-----------|---------|----------|---------|---------|---------|-----------|
| S_i | Minimum | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| | Average | 0.503 | 0.641 | 0.419 | 0.854 | 0.211 |
| | Maximum | 1.000 | 1.000 | 1.000 | 1.000 | 1.000 |
| N_i | Minimum | 0.192 | 0.250 | 0.038 | 0.297 | 0.132 |
| | Average | 0.430 | 0.476 | 0.391 | 0.530 | 0.337 |
| | Maximum | 0.622 | 0.662 | 0.784 | 0.738 | 0.647 |
| Run time | Minimum | 11.160 | 7.010 | 15.100 | 8.170 | 16.980 |
| | Average | 45.559 | 38.904 | 53.109 | 58.638 | 671.745 |
| | Maximum | 110.150 | 125.540 | 116.160 | 158.350 | 3942.270 |
| # of best | | 97 | 47 | 130 | 4 | 362 |

Table 6. Comparison of the algorithms.

different algorithms occurred at different factor combinations. For example, SPT/LATE is the best algorithm when the system is balanced and all other factors except demand mean are at their high levels. The proposed algorithm is again the best one according to the average normalized values for all factor levels. The reason that we used two normalized scaling values instead of the average cost values is as follows. In literature, most of the studies compare the different sequencing rules in terms of the average cost values. This might be misleading since the SPT/LATE rule performs better than the FCFS rule under the tight production conditions, such as high demand variation and long set-up times, which usually result in high production cost values. On the other hand, the FCFS rule performs well under the 'ideal' conditions that result in low production cost values. Taking a simple average of cost values may not give the relative merit of each competing algorithm under different experimental conditions.

Our computational results indicate that when the variety of the products increase, the performance of the system declines. When the product variety increases, the standardization in product design decreases, and the repetitive nature of the system disappears. When we introduce imbalance to the system, the system performance, especially for the FCFS and FCFS-F rules, declines. Allocating different number of kanbans at each stage and using a proper scheduling rule can help to cope with the imbalance in the system. The proposed algorithm performs well when the production system is unbalanced, since a more sophisticated algorithm should be used for the bottleneck machine, which determines the production rate. When the demand variability is high, the performance of the item based rules and the proposed algorithm are better. The family based rules cannot adapt to demand changes quickly due to long production of families. When the number of parts in each family increases, the family based rules have the advantage of reducing the set-up time by producing the items in groups. We also showed that the lower the set-up times, the better the system performance. As the set-up times increase, the JIT characteristic of the system disappears. If the set-up to processing time ratio increases, FCFS and FCFS-F cannot justify the increase in set-up time. When B/I ratio is higher, the normalized values for SPT and SPT-F increases. But, the decrease in the performance of these algorithms is insignificant. Therefore, kanban system performs best when the set-up times are low, the congestion is low and the product variety is low.

The third measure is the run time in seconds as summarized in table 6. The average run time of the proposed algorithm is the highest as expected, since we may have to solve several IP problems, which increases the run time significantly. But, it is still within acceptable limits for such a decision-making problem. The last measure is the number of runs an algorithm gives the minimum total cost among the competing algorithms. In 362 runs, the proposed algorithm gives the minimum cost. The second-best algorithm is FCFS, which gives minimum cost in 130 runs. In sum, although the average run time of the proposed algorithm is considerably high, the average performance is better than the methods commonly used in literature. When we compare the existing methods, we see that the FCFS rule, which is the most frequently used method in literature, is better than other methods, although the SPT/LATE gives a smaller average total cost value than the FCFS. As we pointed out above, this might be due to the safety factor parameter, which is not robust to the selected dispatching rule, and favors the FCFS rule. When the operating conditions are 'ideal' for the repetitive manufacturing, FCFS can give the best results. When the

production setting is imperfect, especially under the tight production conditions, SPT/LATE and the proposed algorithm perform well.

6. Conclusion

In this paper, we studied the kanban systems in a multi-item, multi-stage, multi-period, capacitated modified flowline production environment. We proposed an algorithm that determines both the design parameters of a kanban system such as withdrawal cycle length, kanban size and number of kanbans, and the operational decisions, such as kanban schedules at each production stage, simultaneously. To our knowledge, this is the first study that considers the interdependencies between the design and operational decisions, and evaluate their impact on each other, which we believe can be effectively exploited to improve production efficiency, and lead to substantial cost reductions.

Some of the main results of our study can be outlined as follows. The existing JIT production planning models deal mainly with smoothing production schedules. It is commonly assumed that the final assembly schedule can be propagated back through the system using the FCFS dispatching rule with the instantaneous kanban withdrawal mechanism. We showed that this combination may not be a good policy for all experimental conditions. In contrary, the FCFS rule performs better when the withdrawal cycle lengths are long enough to justify the set-up times. Very few researchers have studied the effects of material-handling frequency on the system performance. We showed that the scheduling policies are not robust to withdrawal cycle lengths, or vice versa. The withdrawal cycle length has a significant effect on average in-process inventories and production rates. As the withdrawal cycle lengths decrease, the in-process inventories decrease significantly, but the back-orders increase. In fact, when the kanban sizes are constant, the total cost curve is convex over withdrawal cycle lengths.

It is a common conjecture that the customer service level, usually expressed in terms of back-order costs and fill rates, can be increased by increasing the maximum inventory level, or the safety stock parameter in the Toyota formula. We showed that this conjecture is only valid for the FCFS rule, which maintains consistent kanban priorities from stage to stage and prevents kanban passing. For other scheduling rules, the best inventory level that maximizes the customer service level may not correspond to the highest possible inventory level. Finally, we compared the proposed scheduling algorithm with the four commonly used sequencing rules in the literature, which are SPT/LATE, FCFS, SPT-F and FCFS-F, and showed that kanban system performance can be significantly enhanced by using a more sophisticated scheduling algorithm instead of the simple dispatching rules. For the future the most promising research direction is to add transportation lead times to the existing model. In that way, a trade-off will appear among different kanban sizes and withdrawal cycle lengths.

Appendix

| | |
|-------------|---|
| a_{ij}^T | kanban size for item (i, j) for the withdrawal cycle length of T |
| B_{ijm}^t | number of back-orders of item (i, j) in withdrawal cycle t at stage m |
| b_{ijm} | unit back-order cost of item (i, j) at stage m |
| D_{ijm}^t | demand for item (i, j) at stage m in withdrawal cycle t (number of kanbans) |

| | |
|-----------------|---|
| d_{ij}^t | actual demand for item (i, j) at withdrawal cycle t |
| d_{ijm}^u | updated demand for item (i, j) at stage m |
| FG_{ij}^t | remnants of item (i, j) at withdrawal cycle t |
| h_{ijm} | unit inventory holding cost of item (i, j) at stage m |
| I_{ijm}^t | in-process inventory of item (i, j) at the beginning of withdrawal cycle t at stage m |
| MAXINV $_{ijm}$ | maximum inventory level of item (i, j) at stage m |
| MS $_{ijm}^T$ | maximum delay of item (i, j) at stage m |
| n_{ijm}^T | number of kanbans for item (i, j) at stage m for the cycle length of T |
| p_{ijm}^u | updated processing time of item (i, j) at stage m |
| p_{ijm} | actual processing time of item (i, j) at stage m |
| $Q_{ijm}[s]$ | number of kanbans of item (i, j) at stage m that are backlogged for s withdrawal cycles |
| size $[i]$ | number of parts in family i |
| S_{ijm}^t | in-process inventory of item (i, j) at the end of withdrawal cycle t at stage m |
| SCH $_{ijm}^t$ | number of item (i, j) scheduled at stage m at withdrawal cycle t |
| UNS $_{ijm}^t$ | number of item (i, j) that remains unscheduled at stage m at withdrawal cycle t |
| w_{ijm} | probabilistic weight of item (i, j) at stage m |

References

- AKTURK, M. S. and BALKOSE, H. O., 1996, Part-machine grouping using a multi-objective cluster analysis. *International Journal of Production Research*, **34**, 8, 2299–2315.
- AKTURK, M. S. and ERHUN, F., 1999, An overview of design and operational issues of kanban systems. *International Journal of Production Research*, **37**, 3859–3881.
- BERKLEY, B. J., 1992, A review of the kanban production control research literature. *Production and Operations Management*, **1**, 393–411.
- CHANG, T.-M. and YIH, Y., 1994a, Generic kanban systems for dynamic environments. *International Journal of Production Research*, **32**, 889–902.
- CHANG, T.-M. and YIH, Y., 1994b, Determining the number of kanbans and lotsizes in a generic kanban system: a simulated annealing approach. *International Journal of Production Research*, **32**, 1991–2004.
- CHANG, T.-M. and YIH, Y., 1998, A fuzzy rule-based approach for dynamic control of kanbans in a generic kanban system. *International Journal of Production Research*, **36**, 2247–2257.
- CO, H. C. and SHARAFALI, M., 1997, Overplanning factor in Toyota's formula for computing the number of kanban. *IIE Transactions*, **29**, 409–415.
- HUANG, C. C. and KUSIAK, A., 1996, Overview of kanban systems. *International Journal of Computer Integrated Manufacturing*, **9**, 169–189.
- HUANG, P. Y., REES, P. L. and TAYLOR III, B. W., 1983, Simulation analysis of the Japanese just-in-time technique (with kanbans) for a multiline, multistage production system. *Decision Sciences*, **14**, 326–344.
- HUM, S. H. and LEE, C. K., 1998, JIT scheduling rules: A simulation evaluation. *Omega*, **26**, 381–395.
- KARMAKAR, U., 1993, Manufacturing lead times, order release and capacity loading. In *Handbooks in OR & MS*, Vol. 4, eds S. C. Graves, A. H. G. Rinnooy Kan and P. H. Zipkin (Amsterdam: Elsevier), pp. 287–329.
- KARMAKAR, U. and KEKRE, S., 1989, Batching policy in kanban systems. *Journal of Manufacturing Systems*, **8**, 317–328.

- LEE, L. C., POO, A. N. and SEAH, K. H. W., 1993, Periodic pull: a modified approach to just-in-time production. *International Journal of Computer Integrated Manufacturing*, **6**, 186–190.
- PHILIPOOM, P. R., REES, L. P. and TAYLOR III, B. W., 1996, Simultaneously determining the number of kanbans, container sizes, and the final-assembly sequence of products in a just-in-time shop. *International Journal of Production Research*, **34**, 51–69.
- RAGATZ, G. L. and MABERT, V. A., 1984, A simulation analysis of due date assignment rules. *Journal of Operations Management*, **5**, 27–39.
- SARKER, B. R. and BALAN, C. V., 1999, Operations planning for a multi-stage kanban system. *European Journal of Operational Research*, **112**, 284–303.
- SAVSAR, M., 1996, Effects of kanban withdrawal policies and other factors on the performance of JIT systems—a simulation study. *International Journal of Production Research*, **34**, 2879–2899.
- TAKAHASHI, K., 1994, Determining the number of kanbans for unbalanced serial production systems. *Computers and Industrial Engineering*, **27**, 213–216.
- YAVUZ, I. H. and SATIR, A., 1995, A kanban-based simulation study of a mixed model just-in-time manufacturing line. *International Journal of Production Research*, **33**, 1027–1048.