Scheduling and Order Acceptance for the Customized Stochastic Lot Scheduling Problem

Nicky van Foreest, J Wijngaard, Taco van Der Vaart

Scheduling and Order Acceptance for the Customized Stochastic Lot Scheduling Problem

Journal:	*International Journal of Production Research*
Manuscript ID:	TPRS-2008-IJPR-0190.R1
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	19-Aug-2008
Complete List of Authors:	Van Foreest, Nicky; University of Groningen, Faculty of Economics and Business Administration Wijngaard, J; University of Groningen, Faculty of Management and Organisation Van der Vaart, Taco; RuG, Faculty of Economics and Business Administration
Keywords:	MAKE TO ORDER PRODUCTION, SIMULATION, DUE-DATE SCHEDULING
Keywords (user):	slsp

International Journal of Production Research
Vol. 00, No. 00, 00 Month 200x, 1–15

RESEARCH ARTICLE

Scheduling and Order Acceptance for the Customized Stochastic Lot Scheduling Problem

Nicky D. van Foreest, Jacob Wijngaard, Taco van der Vaart
Faculty of Economics and Business Administration,
University of Groningen,
P.O. Box 800, 9700 AV Groningen, The Netherlands
(*Received 00 Month 200x; final version received 00 Month 200x*)

This paper develops and analyzes several customer order acceptance policies to achieve high bottleneck utilization for the customized stochastic lot scheduling problem (CSLSP) with large setups and strict order due dates. To compare the policies simulation is used as main tool, due to the complicated nature of the problem. Also approximate upper and lower bound for the utilization are provided. It is shown that a greedy approach to accept orders performs poorly in that it achieves low utilization for high customer order arrival rates. Rather, good acceptance and scheduling policies for the CSLSP should sometimes reject orders to create slack even when there is room in the schedule to start new production runs. We show that intelligently introducing slack enables to achieve high utilization. One particularly simple policy is to restrict the number of families present.

Keywords: Make-to-order production, order acceptance policies, customized stochastic lot scheduling problem, SLSP, simulation.

1. Introduction

Scheduling the production of multiple standardized items on a single machine with limited capacity and significant setup times under random demand is a classic operations research problem. The setups are required to switch production from one type of product to another, while at the same time any time spent on setups wastes scarce capacity.

One approach to reduce the fraction of time spent on setups is to combine individual orders with similar characteristics into *production runs* of a single *product family* such that *intra-family* setups are relatively short while *inter-family* setups are relatively long. Hence, production efficiency increases by spreading the inter-family setup time over the orders within one run. However, the problem becomes now to determine the size of these runs and the production sequence.

In situations that allow for make-to-stock production, for example when products are sufficiently standard, family runs can be reasonably large since the stock serves to decouple production from demand. Here the production control aims at using the *inventory* as efficiently as possible to meet demand.

However, many production situations, for instance when final products are highly customer specific, do not allow to use stock as a means to decouple demand from production. Rather than make-to-stock, the production policy is now *make-to-order*. The additional aspect of make-to-order considerably complicates the lot scheduling problem, as now not only the utilization expressed as the fraction of

Corresponding author: n.d.van.foreest@rug.nl

ISSN: 0020-7543 print/ISSN 1366-588X online

DOI: 10.1080/00207540xxxxxxxxx
http://www.informaworld.com

orders served should be acceptable, also the due dates of the customers should be met. We refer to this situation as the *customized products stochastic lot scheduling problem* (CSLSP) with strict due dates, c.f. Winands *et al.* (2005).

One of the most interesting aspects of the CSLSP is to try to strike a balance between the conflicting internal and external objectives of the production process. On the one hand, since the loss of production time due to inter-family setups is significant, quoting long lead times becomes tempting: the longer the lead times, the more potential to combine orders into long family runs, hence the smaller the setup loss. Moreover, long lead times act as a buffer to smooth demand fluctuations. As such the lead times actually provide the *advance demand information* that enables to efficiently combine orders. On the other hand, customers require short lead times, immediate acceptance decision, and on-time delivery. Consequently, customer service puts an upper bound on the lead times. Therefore, the main problem for the CSLSP is to design a *customer acceptance control policy* that respects on the one hand buyer-supplier relationships and on the other realizes an acceptable utilization.

In this paper we develop and analyze several order acceptance control policies that appear suitable for the CSLSP. These control policies try to accept orders by positioning the orders into a schedule of finite length, which results from the (finite) lead time of newly arriving orders. If an order cannot be scheduled, it is rejected. Given the difficulty of the problem involved we use simple approximate models and simulation to analyze the system behavior.

We first study a straightforward, *greedy*, control rule that tries to adjoin each arriving order to a run of its family, when present. When such an insertion should occur somewhere in the 'middle' of the schedule rather than at the 'tail', the insertion pushes backward all jobs scheduled behind the 'selected' family run. This is only allowed when the pushed back orders can still be produced in time. In case this fails the greedy rule attempts to start a new family run at the end of the schedule. When the schedule has sufficient room for the required setup and the new order, the greedy rule accepts it. Although this greedy policy is appealing by its simplicity, we show that it is *not robust* in that the realized utilization is quite sensitive to the order arrival rate. In fact, for increasing arrival rates, the utilization first increases, but after reaching a maximum it *decreases*.

To realize more robust performance, in the sense of being less sensitive to variations in the arrival rate, it turns out necessary to anticipate on future orders to combine with already present family runs. We develop and analyze four different modifications of the greedy policy to incorporate the required anticipation effect in the acceptance policy.

The development and study of customer acceptance policies for the CSLSP gains further motivation by pointing out that it is actually not so clear to what extent the insights of the make-to-stock case to balance production objectives carry over to the make-to-order case. First, the make-to-order environment uses an order acceptance function whereas the make-to-stock case does not. Second, the make-to-order situation can use advance demand information, i.e., the positive lead-time between the moment of arrival and delivery of an order, to schedule orders. Third, the major performance indicators to evaluate the relevant scheduling policies are more or less 'orthogonal' for the two situations. The performance of policies for make-to-stock situations is mainly expressed in terms of inventory cost and delivery reliability, possibly combined with setup cost, whereas utilization is less important. Assuming that orders are backlogged and that the system is stable the utilization must equal the arrival rate of orders. As, typically, scheduling policies cannot change the arrival rate, they cannot influence this utilization, making this is a less suitable

performance measure. On the other hand, for make-to-order situations the major performance metric is the utilization, i.e., the fraction of accepted orders, and the service level, i.e., the fraction of orders delivered in time. Each of these measures can be directly influenced by the order acceptance policy. Indeed, companies such as a make-to-order tube mill, see, e.g., Schmidt *et al.* (2001), focus on the utilization. Inventory cost, i.e., the cost of producing too early, is not the primary performance measure to evaluate customer acceptance control policies, as this cost is rather insensitive with respect to the choice of acceptance policy. When the utilization is high, the schedule is usually near to full and the most urgent order of each run is delivered just before its due-date. Only less urgent orders in one production run build up inventories. However, producing early is the consequence of forming runs of orders with different due dates. Inventories are as such inevitable for any policy.

The paper is organized as follows. Section 2 summarizes related literature. Section 3 explains the model and the greedy policy. In Section 4 we derive approximate lower and upper bounds on the resource utilization. In Section 5 we evaluate the greedy policy. As alluded to previously, this policy performs poorly when the arrival rate is high. Therefore, we investigate in Section 6 the performance of four other customer acceptance control policies. Section 7 summarizes the results and presents suggestions for further research.

2. Theoretical Background

The make-to-stock version of lot scheduling has received much attention in the literature. Elmaghraby (1978) provides an overview of the case with deterministic demand, i.e., the economic lot scheduling problem (ELSP), and Sox *et al.* (1999)) treats the case with stochastic demand, i.e., the SLSP. More recent overviews are provided by Brander (2006) and Winands *et al.* (2005)

On the other hand, there is hardly any literature on the pure make-to-order case of the SLSP, i.e., the customized products SLSP. Schmidt *et al.* (2001) and Bertrand *et al.* (1990) study tube mills in which the make-to-order property is the consequence of high variety in the last production stages and cyclic schedules control the mill. However, they do not discuss in detail the applied order acceptance policies. Closest related is the work of ten Kate (1994, 1995) and Wester *et al.* (1992) who compare order acceptance based on only aggregate information with order acceptance that is fully integrated with scheduling. Van der Vaart and Wijngaard (2007) consider the multi-machine situation and investigate the possibility to dedicate machine capacity to specific families and reduce the multi-machine case to the one-machine case. Kim and Oyen (2000) and Baker (1999) concentrate on minimizing maximal lateness, but do not consider rejection as a strategy to meet order due dates.

Polling models may seem an attractive approach to tackle the CSLSP, see, e.g., Federgruen and Katalan (1999) and Winands *et al.* (2005). Also interesting is the random polling model with setups analyzed by Kleinrock and Levy (1988). A basic assumption in these polling models is that the queue may have any length whereas in our case the queue length is finite due to the committed due dates. Takagi (1991) and Kim and Oyen (2000) study polling models in which the queue length for each family is bounded. In our case the due date of each order may be interpreted as putting a bound on the total queue length, rather than on the queue length of each family. The former bound introduces interference between the queues of each family, thereby considerably complicating the analysis.

Other queueing models in which the arrival and service rate depend on the work load (or the queue length), see e.g. Bekker (2004), also do not cover the problems

at hand as now the acceptance depends on the *contents* of the queue rather than just the *length*.

There is also some research on hybrid situations (a combination between make-to-order and make-to-stock), e.g., Soman *et al.* (2004), but because of the possibility of building inventory in some products, these situations resemble more the make-to-stock situation than the make-to-order situation.

We conclude that the CSLSP with setup times and strict due dates is theoretically and practically interesting, but no substantial model-oriented research on the different aspects of this situation has been carried out as yet.

3. Model and Performance Measures

In Section 3.1 we introduce the simplest production system that features all the problems of the CSLSP with strict due dates. Then, in Section 3.2 we present the greedy control policy, according to which jobs will be accepted and released into the system. Section 3.3 defines the performance measures of relevance.

3.1 *The Production System*

A single server resource receives orders arriving at rate λ in accordance to an independent stationary Poisson process $P(\lambda)$. Each arriving order belongs to one of N product families with equal probability. Let A_i denote the arrival time of order i and $d_i = A_i + h$ its due date; the lead time h is the same for all orders. Each order should be served before or at its due date.

The resource accepts or rejects orders according to some order acceptance control rule, to be described below. When the resource is busy at an arrival epoch, it queues any accepted order. The resource can serve one order per time unit, and delivers every accepted order before its due date. As such the resource behaves as a single server queue with maximal waiting time h. Between jobs of different families the resource inserts a setup of duration s.

For later reference we use the shorthand (N, h, s) to characterize the production system by its main parameter values. A (family) run is a set of consecutive jobs of the same family.

We assume that the arrival rate is such that the resource is heavily loaded if orders do not combine into family runs, that is, when a setup separates every two orders whether the orders are of the same family or not. In such cases the amount of work arriving per order is one unit of serve plus a setup, i.e., $1 + s$, so that the assumption on λ can be formulated as

$$\lambda(1 + s) > 1. \tag{1}$$

3.2 *The Greedy Acceptance Policy*

Now that we have introduced the production system, we turn to describing the greedy acceptance policy.

The greedy policy contains two order acceptance steps: the *combine* step and the *spawn* step. In trying to accept a newly arriving order the resource first applies the combine step. If this step rejects the order, the resource tries the spawn step. If the order is also refused by this step, the resource rejects the order altogether.

To explain the acceptance steps, we refer to Table 1. The first step, i.e., *combine*, aims at reducing the fraction of setups by trying to combine each order with the

last run of the already accepted orders of the same family. For example, suppose a 'red' order arrives. The schedule clearly contains two runs of the 'red' family, to wit, at positions 1, 2, and 10. Since the schedule has two free places, the resource accepts the new red order and schedules it at position 11, thereby enlarging the second run of the red family. Suppose that, instead of being red, the new order is blue. Then the resource tries to join the order with the run of its 'kin' by scheduling it at position 7, shifting the setup to 8, and the already present orders to positions 9 to 11. This is considered attractive (since it increases the effective processing time of the server) as long as the shifted orders can be produced in time. However, if one of the shifted order will be late, it is not allowed to insert this new order at position 7. Now the resource applies the *spawn* step by trying to adjoin the order to the end of the schedule so that it 'spawns' a new generation of its family. This acceptance will necessarily introduce a setup between the last family and the new order. Hence, the resource schedules the just arrived job at position 12 and insert a setup at position 11. Thus, to accept such an order the schedule should provide room for the setup and one order, as is the case in the instance of Table 1. When there is only one place left, which would be the case if position 11 would be occupied by a red job, the setup would shift the new order past its due date. When this happens, the spawn step also rejects the order.

It is of interest to observe that the greedy policy not only accepts orders, but also *schedules* accepted orders. In this sense, it would be perhaps more appropriate to denote such a policy as an acceptance and scheduling policy. Furthermore, within each run the combine step maintains the earliest due date first order, c.f. Baker (1999) for some insights into the consequences for optimality.

1	2	3	4	5	6	7	8	9	10	11	12
R	R	s	B	B	B	s	G	s	R		

Table 1. *An instance of a schedule of accepted orders. The symbols 'R', 'G', and 'B', refer to order colors, 's' to setup. The size of the schedule is 12 positions.*

3.3 *Performance Measures*

As explained in the Introduction, the utilization is the primary performance measure of interest, as by assumption all accepted orders are served before their due dates. The *utilization* u is the fraction of time the server is working on jobs. Let $N(t)$ and $S(t)$ denote the number of orders and setups served during $[0, t]$, then, since the service time per order is 1,

$$u = \lim_{t\to\infty} \frac{N(t)}{t} = \lim_{t\to\infty} \frac{N(t)}{N(t) + S(t)s}. \tag{2}$$

The second equality follows from assumption (1), as the idle time is assumed negligible.

We are also interested in the long term *acceptance probability* p. Since the resource serves every accepted order, the number of accepted orders equals the number of served orders $N(t)$. Hence,

$$p = \lim_{t\to\infty} \frac{N(t)}{\lambda t}. \tag{3}$$

Combining (2) and (3) gives

$$u = \lambda p \tag{4}$$

4. Approximate Bounds on the Utilization

It appears to be exceedingly difficult to provide an exact mathematical characterization of the dynamical behavior of the CSLSP, c.f.,Winands *et al.* (2005). We therefore settle at deriving approximate lower and upper bounds of the utilization. We derive these estimates under the assumption (1), i.e., that the combined workload of orders and setups is in excess of the capacity of the resource. These bounds are of independent interest also, as the difference between these bounds provides an estimate for the potential increase in utilization offered by the combine step.

4.1 *Lower Bound*

To obtain an approximate lower bound of the utilization, we propose, as an extreme case, to make the combine step, explained in Section 3.2, so strict that it only accepts orders at the tail of the schedule, and never 'in the middle'. In this case the queueing policy reduces to FIFO and family runs only originate at the tail of the queue. This assumption minimizes the average run length, hence maximizes the average number of setups present in the schedule.

To compute the resulting run length, proceed as follows. Suppose that at moment t the resource starts a service, the last order is red, and the schedule is full. The next order to be accepted must necessarily arrive after $t+1$. Moreover, when this next job belongs to another family it must arrive after $t+s+1$. Thus, when no red order arrives in the interval $[t+1, t+s+1]$ any order arriving after $t+s+1$ will be accepted in the schedule. Now

$$\begin{aligned} q &= P(\text{The next order belongs to another family}) \\ &= P(\text{no red arrival during}[t+1, t+s+1])P(\text{next order is not red}) \\ &= e^{-\lambda s/N}\frac{N-1}{N}. \end{aligned}$$

Since the run length of a family is geometrically distributed with probability $1-q$, the expected run length $R = 1/q$. Observe that a setup splits every two runs, so that the minimal production efficiency becomes in analogy to (2),

$$u_{\min} \approx \min\left\{\lambda, \frac{R}{R+s}\right\} = \min\left\{\lambda, \frac{N}{N+s(N-1)e^{-\lambda s/N}}\right\}, \tag{5}$$

as the minimal utilization can never exceed the total arrival rate λ of orders.

4.2 *Upper Bound*

At the other extreme, the combination gain is maximal if the combine step would accept every order and simply ignore whether orders will shifted past their due dates. As a consequence there will be at most one run of each family present in the schedule.

To begin, observe that since every family run starts with a setup, the expected number of setups present in the schedule should equal the expected number of

present families. To compute this we proceed as follows. It is clear that the initial Poisson arrival process $P(\lambda)$ is thinned by the acceptance policy. Now we approximate this thinned stream as a Poisson arrival process $P(\lambda p)$ as with probability p the order is accepted. Since on the long run each of the above policies is unbiased towards family acceptance, the arrival process of a single family must then be well approximated as $P(\lambda p/N)$. The probability that some family i is present equals the probability q that the interarrival time T_i between two *accepted* orders of family i is less that the length of the schedule, thus $q = P(T_i \leq h) = 1 - \exp(-h\lambda p/N)$. As a second approximation we assume that the presence of some orders is of no consequence to the presence of orders of other families. From this, then, it follows that the number of families in the schedule is binomially distributed with parameters N and q, the expectation of which is Nq. The operation is most efficient when the schedule is full, so that of the h positions in the schedule Nqs are consumed by setups, and the rest by orders. Consequently, the maximal utilization efficiency $u_{\max} = \min\{\lambda, (h - sNq)/h\}$, as the utilization can never exceed λ, that is

$$u_{\max} = \min\left\{\lambda, 1 - \frac{sN}{h}\left(1 - e^{-h\lambda p/N}\right)\right\}.$$

On the other hand, by (4), we have that $u = \lambda p$. Combining these two observations into a single equation leads to a fixed point equation for p:

$$\lambda\, p = \min\left\{\lambda, 1 - \frac{sN}{h}\left(1 - e^{-h\lambda p/N}\right)\right\}. \tag{6}$$

That is, for given (reasonable) values of λ, N, h and s we solve (6) for p, and set $u_{\max} = \lambda p$. We occasionally write $u_{\max}(N, h, s)$ to express explicitly the dependence on the parameters N, h, and s.

4.3 *Combination Potential*

A reasonable estimate for the potential increase in utilization due to the the combine step is the difference between (6) and (5), i.e., $u_{\max} - u_{\min}$. We refer to this difference as the *combination potential*. Figure 1 gives an impression of the combination potential for various values of h, N and s. Obviously, the difference is considerable, thereby making a detailed investigation of an optimal acceptance policy interesting.

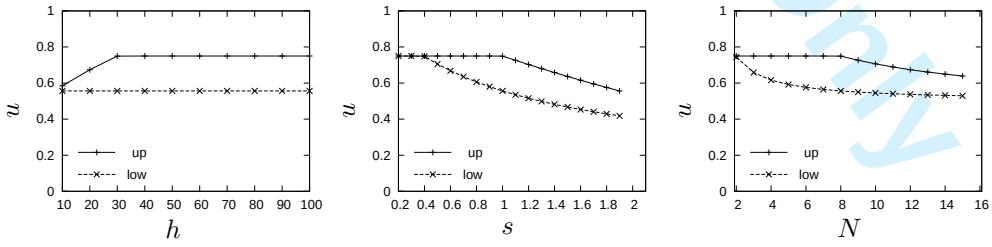

Figure 1. *The upper and lower bounds on u as a function of lead time h, the number N of product families, and setup time s, respectively. In all cases $\lambda = 0.8$. When N, s or h do not vary, they take the values 8, 1 and 30.*

By intuitive reasoning we derive that the combination potential should be a decreasing function of N for N sufficiently large: there are simply too many product families to form runs of interesting length. Also, when s is small the gain in forming

combinations is small. These observations find further support from the following limits of (5) and (6):

$$\lim_{N\to\infty} u_{\max} = \lim_{N\to\infty} u_{\min} = \frac{1}{1+s},$$

where the left equality follows from an expansion of (6) up to first order in N, and

$$\lim_{s\to 0} u_{\max} = \lim_{s\to 0} u_{\min} = 1.$$

5. Evaluation of the Greedy Policy

To evaluate the performance of the greedy policy we use simulations. The reference situation is $N = 8$ product families, setup time equal to $s = 1$, and lead time $h = 30$. The length of simulations is 100,000 time units, making the results reliable up to ± 0.001.

Figure 2 shows some of the characteristic results for the utilization as a function of the arrival rate λ. The left panel compares the utilization for the case $(8, 30, 1)$ to its approximate lower and upper bounds, c.f., (5) and (6). Clearly, we see that by using the greedy control the utilization is *not* an increasing function of the the arrival rate λ. In fact, for high arrival rates the greedy policy performs hardly better than the approximate worst case behavior of (5).

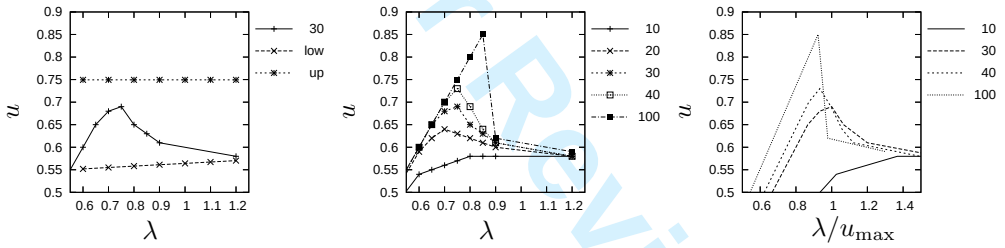

Figure 2. *The utilization as a function of the arrival rate λ for the greedy policy for $N = 8$ and $s = 1$.*

To analyze this substantial drop in performance in a larger context we varied the lead time h from 10 to 100. The middle panel presents the results. Apparently, for all values of h we see worst case behavior for large values of λ.

It appears that when λ is large an unacceptably large fraction of time is waisted on setups. The reason is that the combine step cannot accept orders in the 'middle of the schedule' to form long runs, because, roughly speaking, the schedule is full with short runs. These short runs result from the fact that for high values of λ any room originating at the tail of the schedule is rapidly filled by either orders or setups. As a consequence most accepted orders have no or just small slack between the delivery date and the due date, thereby making it impossible for the combine step to squeeze in a new order. Actually, this type of acceptance behavior is at the base of the lower bound (5).

One direction to mitigate the problem of only forming small runs is to extend the lead time h. Indeed, the graphs in the middle panel show that this is effective, however, only up to some point. When λ is large, relative to $u_{\max}(h)$, the utilization drops to worst case. To clarify this further, we show in the right panel the same results as in the middle panel but now as a function of $\lambda/u_{\max}(h)$. We see that the larger h, the more sensitive is the utilization to λ in the neighborhood of $u_{\max}$. From this figure we infer that $u_{\max}$ is a critical threshold for the arrival rate.

Simulations with various combinations (N, h, s) show similar behavior. This leads us to conclude that the performance of greedy control is quite unsatisfactory and that long lead times do not solve the problem. Consequently, trying to maximize utilization for the short term is far from optimal. It may better to accept short term penalties in the form of rejecting to orders to achieve high long term performance. In other words, anticipating future combination possibilities with as yet not available orders might lead to better results than the greedy policy, which focuses on immediate customer satisfaction. This is the topic of the next section.

6. Four Alternative Acceptance Policies

The analysis in the previous section shows that the greedy policy fails because the spawn step accepts orders too easily when the arrival rate is high. To repair this flaw in the acceptance policy we propose to extend the spawn step with additional rules, which we present in section 6.1. The resulting heuristics for order acceptance aim to protect the combination potential that the combine step tries to exploit. We evaluate these heuristics in section 6.2 to see whether the adapted acceptance policies strike a good balance between the combine and the spawn step.

6.1 *Improved Acceptance Policies*

The problems with the greedy policy as outlined above leads us to investigate four alternative acceptance policies that enable to accept a larger fraction of orders in the combine step by making the rules at the spawn step stricter. Thus, all four policies keep the combine step but differ in the precise rules of the spawn step. Here the description of these steps is rather informal. In Section A we present the details.

The spawn step of the *setup* policy is the simplest. An order is accepted at the tail of the schedule if the greedy policy would accept it and it the number of setups present in the schedule at arrival time is less than or equal to some threshold S.

The spawn step of the *time distance* policy only accepts an order at the tail if the order can be delivered ch time units before its due date. This creates a *distance slack* ch. We infer that there is some optimal $c \in [0, 1]$, which we motivate as follows. The larger the distance slack ch, the more potential for future combinations. On the other hand, if $c = 1$ the spawn step will only start a new run when the schedule is empty, so that the fraction of idle time of the resource is no longer negligible. Observe that if $c = 0$ the slack policy reduces to the greedy policy.

The *generation distance* policy maintains a minimal distance between two subsequent generations of the same family. The spawn step in this case only accepts an order if a new family run does not get closer than the *generation slack* dh to the start of the last run of the same family. If $d = 1$, there are no two runs of the same family in the schedule; if $d = 0$ we have the greedy policy again.

The *cyclic* policy protects the distance in the same way as the generation distance policy, but also guarantees a *fixed cyclic order*. Here the spawn step tries to insert an order at some position p between two adjacent runs if the fixed cycle order requires this. When this is not possible because an accepted job will be late, the spawn step accepts the order to form a new run at the end of the schedule provided the time between the preferred position p and the end of the schedule is less than dh. Note that the case $d = 0$ does not reduce the cyclic policy to the greedy policy, as the cyclic spawn step is allowed to insert new family runs in the middle of the schedule, rather than only at the end.

All four rules provide the possibility to anticipate on future combination possibilities. Time distance control anticipates directly by adding a new run only if the order starting that run has some slack, controlled by the parameter c, between scheduled delivery time and committed due time. The other policies anticipate indirectly by restricting the number of runs in total (setup) or per family through a proper choice of d (generation and cyclic).

6.2 *Evaluation*

In this section we investigate the performance of the above four policies by simulation. As in Section 5 the reference case is $(8, 30, 1)$. Appendix B provides results for various other combinations of (N, h, s). The results shown there support the observations made here.

Figure 3 contains the graphs of the utilization u for the policies and various choices of the controls as a function of $\lambda \in [0.5, 1.3]$. We include as a reference the performance of the greedy policy as the cases $c = 0$ and $d = 0$ for time and generation distance, respectively.

The first observation is immediate: the striking improvement in utilization achieved by the four policies as compared to the greedy policy. For instance, around $\lambda = 0.8$ the utilization increases by approximately 10%. Secondly, the adapted policies are much more robust to changes in the arrival rate, provided the control parameters are chosen appropriately. Thirdly, there is not a single value of c, or S, for the time, or setup, policy that gives (near to) optimal performance for the entire range of λ. On the other hand, generation and cyclic control work (nearly) optimal with maximal slack, i.e., $d = 1.0$.

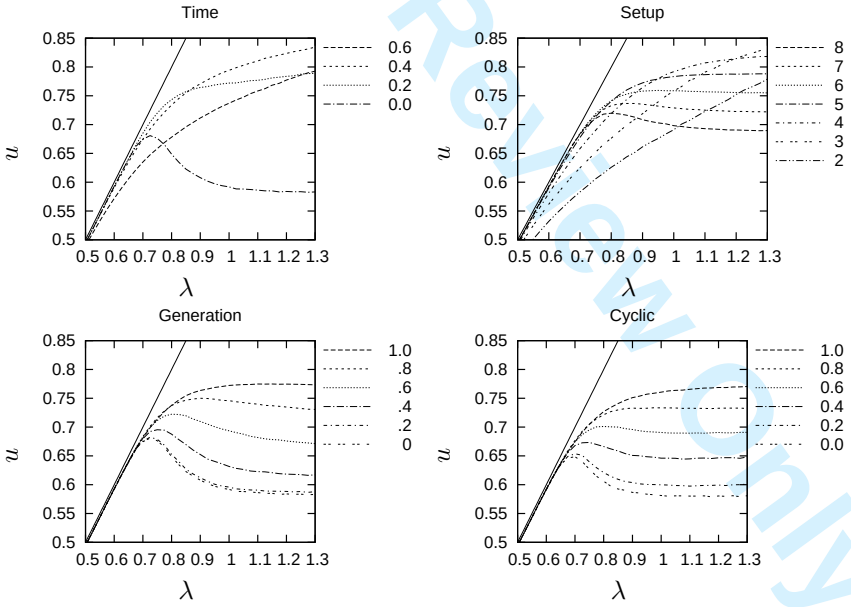

Figure 3. *The utilization as a function of the arrival rate λ for the four acceptance policies with various values of the protection factors for $(N, h, s) = (8, 30, 1)$. The straight line $y = x$ represents the zero loss situation; hence the difference between the graphs for the policies and this line shows the fraction of lost orders.*

A particularly relevant range for λ, from the practical point of view, is around $u_{\max}$, which is approximately 0.75 for the present case $(8, 30, 1)$. By varying λ from 0.6 to 0.8 it appears that the (near to) optimal control values are $c = 0.2, d = 1, S = 6, d = 1$ for the time, generation, setup, and cyclic policy, respectively. Figure 4 shows the utilization for the four policies with the above values for the control

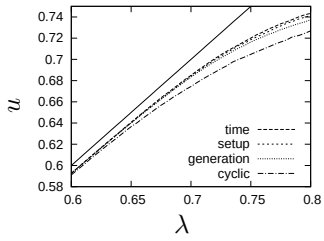

Figure 4. *The utilization as a function of the arrival rate λ for the time policy with $c = 0.2$, generation with $d = 1$, setup with $S = 6$, and cyclic with $d = 1$.*

parameters. The graphs show that time distance control leads to the highest utilization of all four policies, setup and generation are very close, except for relatively high values of λ, and cyclic performs worst. These results support the claim that the inflexibility resulting from producing according to a fixed cycle is somewhat costly. Thus, cyclic control appears to be only interesting if production regularity is important, which is typically the case if the fixed cycle is communicated to the customers so that they might be willing to adhere to this cycle and accept longer lead times to compensate for the performance loss. In case it is not possible to develop a close relationship with customers, the other three policies appear more interesting.

As a second step we investigate the dependence of the utilization u for the different policies for different values of h, N and s, just as in Figure 1. For all combinations we solve (6) for the largest λ such that the acceptance probability $p \leq 1$. Clearly, this critical arrival rate equals $u_{\max}(N, h, s)$. For time and generation distance, and cyclic, we take $c = 0.2$ and $d = 1$. The setup threshold S must of course depend on N and h. Some experimentation showed that taking S as the integer part of $N \exp(u_{\max} h/N) - 1$, compare the derivation of (6), is (near) to optimal.

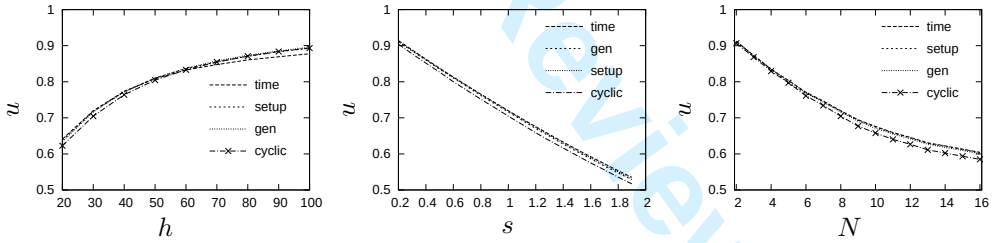

Figure 5. *The utilization as a function of N, h and s while the arrival rate $\lambda = u_{\max}$ varies according to the choice of (N, h, s). The top graph (with index* up*) corresponds to $u_{\max}$.*

In the plots we see that $u_{\max}$ captures the trends of the simulations very well; the actual results are about 5% less than $u_{\max}$.

Generation distance and setup perform well over the entire parameter regions. Interestingly, for larger lead times, time distance with $c = 0.2$ has a lower utilization than the other policies. Other simulations (the explicit results are not included) show that $c = 0.2$ is too small for large lead times. By taking c larger for larger h time distance gives the same results as generation and setup. It would be of interest to find analytical methods to obtain good estimates for c as a function of h. Finally, the cyclic policy never achieves better utilization than the other policies with good control values, but when N is small, or h large, it is nearly as good.

7. Summary and Conclusions

In this paper we develop and analyze customer order acceptance policies to achieve high bottleneck utilization for the customized SLSP with setups and strict order

due dates. To benchmark the performance of the policies we derive an approximate upper and lower bound for the utilization. The former bound is also an upper bound on the arrival rate if zero loss if required. To compare the policies we use simulation, as a detailed analysis appears very complicated.

The order acceptance policies apply two subsequent steps to accept and schedule orders. The combine step, which is the same for all policies, tries to adjoin arriving orders to runs of the same family type. This is acceptable as long as orders that are pushed back due to the insertion will not be late. The details of the spawn steps differ among the policies.

The spawn step of the greedy policy accepts orders whenever possible. It turns out that this policy performs poorly: when the order arrival rate is high the utilization decreases to worst case. The reason is that this greedy strategy does not introduce slack in the schedule, so that future orders cannot form runs of considerable size. This observation leads us to infer that a good spawn step should sometimes reject orders even when there is room at the tail of the schedule, with the aim to protect the combination potential for future arrivals.

We then compare four policies that introduce the required slack in different ways. The time distance policy rejects orders that cannot be served within $(1-c)h$ time units of their due date, thereby maintaining an amount ch of slack in the schedule. The generation distance policy introduces slack in the schedule by preventing runs of the same family to get closer than dh time units. The setup policy only accepts new family runs when the number of setups present is less then some threshold. Finally, the cycle policy is a variant of the generation distance policy that also tries to maintain a fixed production cycle.

Our main observation is that the time distance, generation distance and setup policy achieve high utilization, provided the control values are chosen properly, while the cyclic policy performs somewhat less. Moreover, the performance of these policies is not sensitive to variations in the arrival rate, in contrast to the greedy policy. Overall, we conclude that intelligently adding slack is good strategy to exploit the combination potential.

We envisage various ways to extend the present work. Firstly, we conjecture that introducing slack into the schedule is also necessary in more realistic, and hence more complicated, make-to-order situations with large setup times at a bottleneck resource. Simulations with non-identical product families provide some support for this conjecture already.

Our assumption that the resource is completely reliable enables us to use deterministic policies to schedule and accept orders. Whether such policies perform sufficiently well in the presence of random fluctuations in for instance processing and setup times remains for further research.

A general investigation of the influence of the policies on fairness when the requirements of the families may differ is also important. Suppose, for instance, that the lead time of one family is twice that of the other families, i.e., $2h$ rather than h. Clearly, this family does not have to compete for the schedule capacity in the interval $[t+h,t+2h]$.

When the orders can be served by multiple resources, rather than one, many new problems come into play. Designing good customer acceptance control policies becomes now very challenging as the policy should also incorporate a production allocation decision, c.f. Van der Vaart and Wijngaard (2007).

References

Baker, K., 1999. Heuristic Procedures for Scheduling Job Families with Setups and Due Dates,. *Naval Research Logistics*, 46 (8), 978–991.

Bekker, R., 2004. Finite Buffer Queues with workload-dependent service and arrival rates. Thesis (PhD). Eindhoven University of Technology.

Bertrand, J., Wortmann, J. and Wijngaard, J., 1990. *Production Control, a Structural and Design Oriented Approach*. Amsterdam: Elsevier.

Brander, P., 2006. Inventory Control and Scheduling Problems in a Single-Machine Multi-Item System. Thesis (PhD). Lulea University of Technology.

Elmaghraby, S.E., 1978. The economic lot scheduling problem ($ELSP$): review and extensions. *Management Science*, 24, pp. 587–598.

Federgruen, A. and Katalan, Z., 1999. Impact of adding a make-to-order item to make-to-stock production systems. *Management Science*, 45 (7), pp. 980–994.

Kim, E. and Oyen, M.V., 2000. Finite-capacity Multi-class Production Scheduling with Setup Times. *IIE Transactions*, 32, 807–818.

Kleinrock, L. and Levy, H., 1988. The Analysis of Random Polling Systems. *Journal of Operations Research*, 36 (5), 716–732.

Schmidt, E., Dada, M., Ward, J. and Adams, D., 2001. Using cyclic planning to manage capacity at Alcoa. *Interfaces*, 31 (3), pp. 16–27.

Soman, C.A., Donk, D.V. and Gaalman, G., 2004. Comparison of dynamic scheduling policies for hybrid make-to-order/make-to-stock production systems with stochastic demand. pp. 527–543.

Sox, C., Jackson, P., Bowman, A. and Muckstadt, J., 1999. A review of the stochastic lot scheduling problem. *International Journal of Production Economics*, 62 (3), pp. 181–200.

Takagi, H., 1991. Analysis of Finite-Capacity Polling Systems. *Advances in Applied Probability*, 23 (2), 373–387.

ten Kate, H., 1994. Order acceptance strategies and lead times. pp. 119–129.

ten Kate, H., 1995. Order acceptance and production control. Thesis (PhD). University of Groningen.

Van der Vaart, J. and Wijngaard, J., 2007. The Contribution of focus in collaborative planning for make-to-order production situations with large setup times. *International Journal of Production Economics*, 108 (1–2), pp. 436–443.

Wester, F., Wijngaard, J. and Zijm, W., 1992. Order acceptance strategies in a production-to-order environment with set-up times and duedates. *International Journal of Production Research*, 30 (6), pp. 1313–1326.

Winands, E., Adan, I. and van Houtum, G., The Stochastic Economic Lot Scheduling Problem: A Survey. , 2005. , Technical report BETA WP-133, Beta Research School for Operations Management and Logistics.

Appendix A. Policy Details

Here we describe the customer acceptance control policies in detail. First we need some notation. A job of family r with lead time h arrives at time t at the resource, requires 1 time unit of service, and introduces s time units of setup if it spawns a new family run. The schedule contains n jobs with scheduled delivery times $t < t_1 < t_2 < \ldots < t_n$ and due dates $d_1 < d_2 < \ldots < d_n$. Obviously, $t_i \leq d_i \leq t_i + h$ for all i. Let f_j denote the family of job j. Define $R(r) = \{i \mid f_i = r\}$, i.e., the set of jobs of family r present in the schedule. Let j_r be the index of the last job of family r, i.e., for all $i \in R(r)$, $t_i \leq t_{j_r}$.

The acceptance procedures we consider in the paper consist of the application of

the following set of subsequent steps.

(1) An order at an empty schedule is always accepted. A setup is inserted if the family of the order is different from the family of the last departure.
(2) Tail check 1: Check whether the order fits in the schedule at the tail. If successful, continue with the next step, otherwise reject the order.
(3) Apply the combine step (to be explained below). If successful, accept the job, otherwise proceed to the next step.
(4) Tail check 2: Check whether a setup and the order fit in the schedule at the tail. If successful, continue with the next step, otherwise reject the order.
(5) Apply the spawn step (to be explained below). If successful, accept the job, otherwise reject it.

The first tail check involves to see whether the order fits in the schedule at all. Suppose it would join the end of the schedule. If its due date $t+h$ is smaller than its potential departure time t_n+1 the order cannot be served in time, nor can it be inserted at any other position in the schedule, as necessarily another job will then be late. Hence, the order must be rejected in this case.

The combine step accepts the arriving order of family r if inserting it at the end of the run of its family will not delay the present jobs too much. In formal terms, the order is accepted whenever $R(r) \neq \varnothing$ and $t_k+1 \leq d_k$ for all k such that $j_r < k \leq n$.

The second tail check precedes the spawn step. As the combine step failed, the insertion of the order will necessarily insert a setup at some position in the schedule. Therefore we first check whether the latest possible departure time of the order is smaller than the due date, i.e., $t_n + s + 1 \leq t + h$.

The details of the spawn step depend on the actual policy. The *setup* policy is particularly simple. Whenever the number of setups present in the schedule, or equivalently, the amount of time spent on setups, does not exceed some threshold S the order is accepted. The *time* distance policy requires the presence of some slack at the end of the schedule in order to let the job spawn a new family run at the end of the schedule. More specifically, accept when the departure time $t_n + s + 1 \leq t + (1-c)h$. When $c = 0$ this case reduces to the greedy policy, as $t+h$ is the due date of the order. Observe that the order with the setup will fit into the schedule, for to arrive at Step 5 it is necessary to pass Step 4. The *generation* distance policy requires that subsequent family runs are not too near to each other. Formally, accept if $R(r) \neq \varnothing$ and $t_{j_r} \leq t + (1-d)h$, or accept as in the greedy policy if $R(r) = \varnothing$.

The details of the *cyclic* policy are slightly more complex than the other policies. In simple terms it tries to maintain a cyclic production scheme. An example serves perhaps best to clarify the rule. Suppose the families in the schedule are ordered as $6, 1, 4$, and an order of family 7 arrives. To maintain a cyclic order we would like to insert this order between the families 6 and 1. In detail, we first look up the job k after which the arriving order of family r should be inserted according to the cyclic scheme, that is, we search for the first job k, starting from the last job in the schedule (which is of family f_n) such that

$$(f_k + N - 1 - f_n) \mod N \leq (r + N - 1 - f_n) \mod N. \tag{A1}$$

Insertion behind job k is allowed if $t_j + s + 1 \leq d_j$ for all j such that $k < j \leq n$. If this insertion fails, the job is adjoined to the tail of the schedule when $t_k \leq t + (1-d)h$.

Appendix B. Additional Figures

Figures B1, B2, B3, and B4 provide further support for the observations made in Section 6. In the first set of systems ((4, 30, 2) and (16, 30, 0.5))) and second set (((8, 15, 0.5) and (8, 60, 2)), respectively, we leave Ns and h/s, respectively, unchanged.

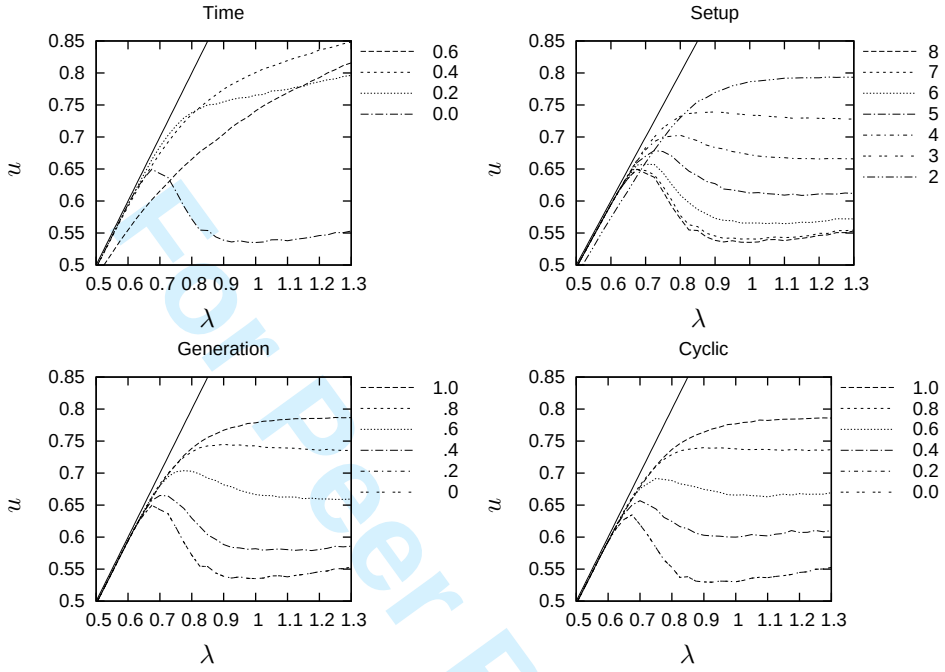

Figure B1. *The effect of the protection factor for the system (4, 30, 2).*

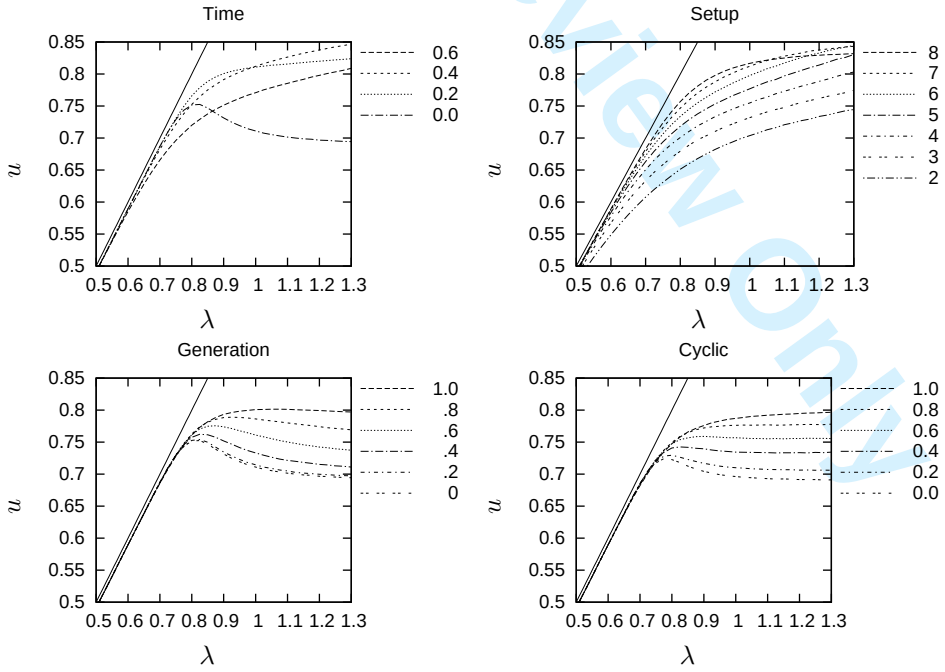

Figure B2. *The effect of the protection factor for the system (16, 30, 0.5).*

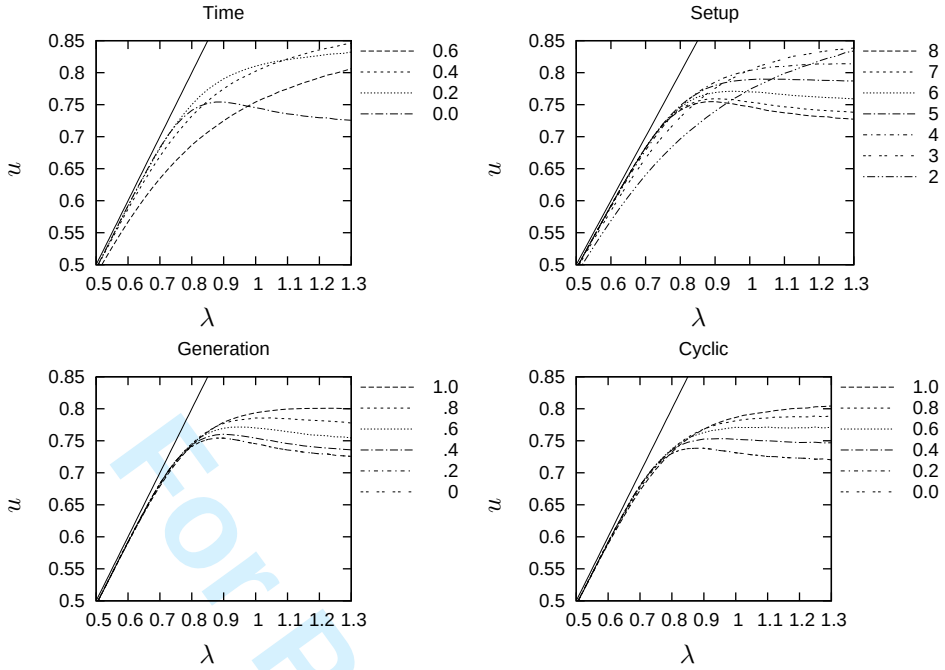

Figure B3. *The effect of the protection factor for the system (8, 15, 0.5).*

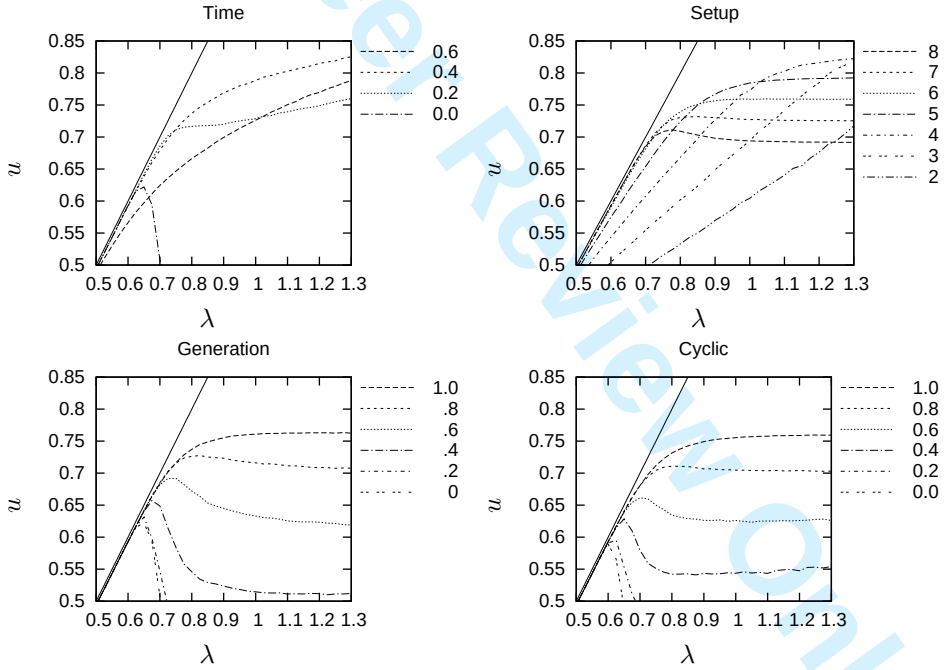

Figure B4. *The effect of the protection factor for the system (8, 60, 2).*