A nonlinear model for optimizing the performance of a multi-product production line

Walid Abdul-Kader a, Ozhand Ganjavi b and Fazle Baki c

a Department of Industrial and Manufacturing Systems Engineering, Faculty of Engineering, University of Windsor, 401 Sunset Avenue, Windsor (ON), Canada, N9B 3P4
b School of Commerce and Administration, Laurentian University, 935 Ramsey Lake Road, Sudbury (ON), Canada, P3E 2C6
c Odette School of Business, University of Windsor, 401 Sunset Avenue, Windsor (ON), Canada, N9B 3P4
E-mails: kader@uwindsor.ca [Abdul-Kader], oganjavi@laurentian.ca [Ganjavi], fbaki@uwindsor.ca [Baki]

Received 21 July 2009; received in revised form 18 November 2010; accepted 29 March 2011

Abstract

This paper examines the measures of performance and in particular addresses the throughput of an automated production line processing multiple products. The line is composed of a sequence of workstations connected in series with finite buffers in between. We explore the effects of buffer size on attenuating the impact of line blocking and starvation that can cause a reduction in the output. Such effects are analyzed through a nonlinear mathematical programming model and the implications are examined. The aim of the model is to achieve the best performance subject to available workstation capacity without overexpenditure on buffer size. Single and multi-objective optimizations are carried out in the paper. A numerical example of a production line with a given configuration of workstations; workstation capacity; and job mix is presented to demonstrate the model and its application. A discussion of the impact of buffer size on maximum throughput is also provided. The paper is concluded with a discussion on the decision-making implications.

Keywords: production line; multi-product; throughput; buffer; makespan; lexicographic optimization

1. Introduction

Globalization of business and commerce is the source of ever-increasing pressure on manufacturers to become lean and competitive. The main thrust of this paper is the creation of a mathematical model to assist manufacturers in their efforts to produce more with the means that are available to them. Increasing the capacity of production facilities by adding more machines or using more efficient machines is often not a solution that is possible to implement in the short or the medium term. Furthermore, the cost of such a decision may at times be prohibitively high. However, the allocation/reallocation of buffer space between workstations to hold the
semi-finished products may turn out to be a move that improves the throughput with relatively less cost associated with such modification or reconfiguration.

The existence of buffers between workstations is aimed at eliminating or reducing the instances of blocking and starvation that can occur due to variation in the production rates caused by random events. Blocking occurs when the buffer after a workstation is full and the subsequent workstation does not drain it fast enough to make room for more products to come in. Starvation occurs when the buffer before a workstation is empty and the upstream workstation does not feed it fast enough to maintain the downstream workstation in operation.

The schematic diagram presented in Fig. 1 depicts a production line with \( m \) workstations buffered with \((m-1)\) buffers. The flow direction is indicated by the arrows.

An integer nonlinear programming model is developed in this paper to optimize various objectives by judiciously adjusting/allocating the buffers of the system. Among the objectives that the model can consider are maximization of throughput, maximization of free time for maintenance and/or other production support-related purposes, minimization of the total production time and minimization of makespan. Another important application would be to find the optimal buffer sizes for a given peak demand, production capacity, total production time or makespan. If needed, this procedure may then be easily extended to minimize the cost of the buffers.

2. Literature review

While the details utilized in this proposed model have been discussed in the literature, there is a lack of a comprehensive model that would consider the issues we try to address. We have encountered some articles that attempt to find the best output for a given buffer size. Issues such as the makespan or the throughput and how they are impacted by buffer size, when the demand is known, are areas that need further development.

Some of the published research is related to buffer allocation. For example, Gershwin and Schor (2000), using a gradient method, provide an efficient algorithm for determining how buffer space should be allocated. Spinellis and Papadopoulos (2000) use a simulated annealing approach to perform buffer allocation. Shi and Men (2003) propose a hybrid Nested Partitions (NP) and Tabu Search (TS) method to arrive at the optimal buffer allocation. Smith and Cruz (2005) utilize an approximation closed-form formula for the M/M/1/K and M/G/1/K queues for buffer allocation problem in series, merge and splitting topologies of finite buffer in networks of queues. Cruz et al. (2010) consider a bi-criteria approach to evaluate buffer allocation and the throughput.
trade-offs in M/G/1/K queuing networks. An approximation method has been used with a special version of a multi-objective genetic algorithm. Lavoie et al. (2009) use a combined discrete/continuous simulation modeling, design of experiments and response surface methodology to optimize a set of transfer lines, with one parameter per machine, for up to seven machines. They report that the last buffer in a production line is more important than the others. Shi and Gershwin (2009) present an algorithm for buffer design with the aim of maximizing the production line profit. A nonlinear programming approach is used in their model and examples of a production line composed of six machines and five buffers are tackled and presented in their paper. Shaaban and Hudson (2008) find that buffers of unequal size placed appropriately between workstations can sometimes even lead to increases in effectiveness. They studied 8 patterns of production lines and determined how to allocate different-sized buffers between workstations and how the allocation depends on the particular conditions of the production facility. Yamamoto et al. (2008) present a production simulator system that consists of a genetic algorithm system and a discrete simulator to decide a buffer size for any flexible transfer line with bypass lines. These papers deal mostly with production lines processing a single product type with a focus on a single performance measure.

Other papers deal with system performance and related issues. Amin and Altiok (1997) study the behavior of a multi-product, multi-stage pull-type production/inventory system operating under a variety of manufacturing control policies. The system investigated consists of multiple stages in series with finite interstage buffer space and a single processor. A setup is involved whenever a switch-over takes place to another product at any stage. The reliability of workstations has not been discussed in this paper. Van Nyen et al. (2005) propose a three-step heuristic to coordinate production and inventory control decisions in an integrated multi-product multi-machine production-inventory system characterized by job shop routings and stochastic demand, setup and processing times. Li and Huang (2005) utilize an “overlapping decomposition” approach to model and analyze the performance of multiple-product manufacturing systems. The accuracy of the model’s output was estimated using a numerical method. Roser et al. (2003) use simulation to study the effect of buffers on the throughput. He et al. (2007) study the output variance and the delivery-time variance for a production line using an approximation method based on Markovian arrival process, and the buffer capacity is from 1 to infinity. In addition to the limitations of simulation and decomposition modeling methods, the above-reviewed papers focused on a single objective where the production line problems are of a multiple objective nature.

Modeling manufacturing systems using heuristics algorithms has some limitations in terms of the model’s development time, convergence and quality of results. These methods are still not easy to explain to decision makers and/or not well accepted in the industry. Based on the above review, we reiterate the contribution and the benefits of implementing a more comprehensive modeling approach to easily address various measures of performance as individual or multiple objectives for a multi-product multi-stage production line and provide answers to many questions that a decision-maker may have in terms of buffer sizes, throughput, makespan, available time for maintenance and other measures used in manufacturing systems.

The remainder of this paper is organized as follows: Section 3 outlines the development of the integer nonlinear model by giving the notations and explaining the constraints and the objective function(s). A numerical example of how the model may be utilized is given in Section 4 along with
discussions on decision-making implications. The paper concludes by addressing the importance of buffers in increasing the throughput and provides some directions for future research.

3. The model

The system under study, as presented in Fig. 1, is an automated serial production line composed of workstations separated with finite buffer spaces between them. Some of the main assumptions in the model include the following. Machines at these workstations are assumed to produce at a known constant rate. The impact of workstation’s random failure is only implicitly incorporated into the model. This can be achieved by inflating the processing time of products by a factor of \(1/A_i\), where \(A_i\) is the long-term availability of workstation \(i\) for \(0 < A_i \leq 1\). The production is carried out in a small number of batches of different products of high volume. This paper deals with a predetermined production sequence. However, due to the small number of product types, the optimization of the production sequence may be explored by examining all permutations of such a sequence. Each workstation consists of one machine that will process one product type at a time, which then becomes the input to the next workstation via the intermediate buffer. For the sake of simplicity, the terms machine, station and workstation are used interchangeably throughout this paper.

Each product type \(j\) is produced in batches of size \(N_j\) and goes through every of the \(m\) workstations. A buffer capacity of \(b_i\) is utilized between workstations \(i\) and \((i+1)\) to lower the frequency of blocking and starvation.

Every item of a product type is passed on to the downstream workstation individually. This is in contrast to the batch transfer environment, where all items of a product are passed on at once, after the batch is completely processed on a workstation. The case of batch transfer has two important characteristics: first, it generally requires a larger capacity for buffers. Second, unless we use the same space for what comes in and what goes out, we need to separate storage around each workstation: one to receive the batch that has arrived and one to accumulate the batch that is to be shipped out. This is a rather inefficient use of space.

Larger buffer sizes have some positive effects on the production throughput. We shall explore this positive effect. If the additional buffer space reduces the total time needed in the workstation, the leftover time may be utilized for productive activities such as preventive maintenance or other useful activities.

3.1. Notation

Before proceeding to construct the elements of the model, below is the notation used for the indices, decision variables and parameters utilized in the proposed model.

**Indices**

- \(m\): Number of workstations in the production line
- \(n\): Number of product types processed on the production line
- \(i \in \{1, 2, \ldots, m\}\): The index for workstations
- \(j \in \{1, 2, \ldots, n\}\): The index for product types
- \(r\): A dummy index
Parameters

\( b_i \) The size of buffer capacity between workstation \( i \) and \((i+1)\) is a system parameter. If the value of \( b_i \) is not fixed in advance, then it would become a decision variable

\( p_{i,j} \) Processing time of one item of product \( j \) on workstation \( i \)

\( s_{t_{i,j}} \) Setup time of workstation \( i \) to process a batch of product \( j \)

\( M_n \) A maximum limit for makespan \( C_{m,n} \). If this limit is not defined in advance, then the relevant constraints (constraints (4)-(8) shown in Section 3.3 below) can be eliminated

\( T \) This system parameter (manufacturing capacity) is the maximum allowable time on any workstation for setup, operation, as well as blocking and starvation

Decision variables

\( N_j \) This is the number of units of product \( j \) to produce. If its value is fixed in advance, it would become a parameter; otherwise, it would be a decision variable

\( d_{i,j} \) The allocated time of workstation \( i \) to batch \( j \) accounting for the setup, processing, starvation and blocking times of this batch. This is the time span from the point when the setup of workstation \( i \) for batch \( j \) starts to the point when the last piece of the batch is moved to the downstream buffer

\( C_{i,j} \) Completion time of product \( j \) on workstation \( i \)

\( C_{m,n} \) Makespan, completion time of last product \( n \) of the mix on last workstation \( m \)

\( Util(i) \) Utilization of workstation \( i = \sum_j (N_j p_{ij} + s_{t_{i,j}})/T \). The value between parentheses in the numerator refers to the operating time, or \( OT \), the time machine \( i \) is busy doing product type \( j \). The total operating time for a workstation would be the time it takes to process all the products

3.2. Objective functions

The proposed model is an integer nonlinear programming model, which may have different objective functions in different scenarios including any of the following five objective functions the management may want to consider:

(a) Maximize throughput \( \Sigma N_j \), given buffer size and manufacturing capacity, \( T \). It should be noted that when we speak of buffer size, we mean \( \Sigma b_i \).

(b) Maximize maintenance time, so Maximize \( \left[ \min_i \left( T - \sum_j d_{i,j} \right) \right] \) or equivalently Minimize \( \left[ \max_i \sum_j d_{i,j} \right] \).

(c) Minimize total production time \( \Sigma \Sigma d_{i,j} \). This would be a helpful measure for costing purposes for cases when machines are leased.

(d) Minimize makespan, \( C_{m,n} \). It is important to know whether the customer can be served within a particular timeframe. \( M_n \) is a maximum limit for makespan \( C_{m,n} \). Thus, Objective (d) can be correctly stated as follows: Minimize Maximum Makespan \( C_{m,n} \).

(e) Minimize the buffer size for a given peak demand \( N_j \forall j \); capacity \( T \); total production time \( \Sigma \Sigma d_{i,j} \); and makespan \( M_n \).
Intuitively, one expects that the lack of any buffer between workstations would have a serious adverse effect on the performance measures discussed above. Providing an increasing amount of buffer space can have a positive effect on the performance; however, at very high levels of buffer size, any additional increase in buffer size is expected to have no or very little impact on the optimal performance measure. This line of thinking leads us to believe that the graph of performance versus buffer size must look like the graph in Fig. 2. This conjectured pattern is examined in more detail in Section 4. This intuitive expectation could be verified once the numerical results are presented. However, the intuitive nature of this expectation can be explained where each additional space will be utilized by the most-needed spot. The fact that most needed spots are served first explains the diminishing contribution of additional buffer. As the space is added gradually, a point will be reached where no spot is left unfulfilled. In addition to this interpretation, many papers dealing with buffer allocation and system performance have found this type of graph. Among others, we mention Buzacott (1971), Hillier and Boling (1966) and Askin and Standridge (1993).

The points below the curve are feasible, but if considered, it would be an inefficient way of utilizing resources, and the points above the curve, are infeasible. Such a pattern of performance has implications in setting up the objective function. Of particular importance is the flat part of the curve at high levels of buffer size. If this pattern prevails, then the maximization of performance, for example maximization of throughput, will have multiple solutions, most of them at excessively high levels of buffer size. There are few approaches to deal with problems of this nature.

One approach is the introduction of a penalty component in the objective function. An example of such a modified objective function would be: Max [performance \(-\varepsilon \) (buffer size)], where \(\varepsilon\) is a very small positive constant. While \(\varepsilon\) is chosen to be a very small value such as 0.000001, its actual choice is expected to have some impact on the level of buffer size in the optimal solution. It should be noted that in cases when more than two objectives are considered, this technique does not work.

Another approach is the use of lexicographic optimization modeling. In such a case, the objective function is Max \([p_1 \text{ (performance)} - p_2 \text{ (buffer size)}]\), where \(p_1\) and \(p_2\) are the priority orders. The optimization problem is solved for the first priority part of the objective (e.g., performance) first. The optimal value of the performance will be added as a constraint when the

---

Fig. 2. Optimal throughput as a function of buffer size.
problem is solved for the second priority level (e.g., the buffer size). One expects that the optimal solution in the second phase would be the point on the graph where the flat part of the curve starts. This is particularly helpful because of the scarcity of available space on the production floor and for inventory management (or for work-in-progress reduction) purposes. One of the good features of the lexicographic approach is the fact that one can find when the flat part of the curve starts. In this region, accepting to lower the optimal performance by a small amount will result in a substantial reduction in buffer size. This can be achieved by a slight reduction of the RHS of the newly introduced constraint in the second phase of the optimization process. In the lexicographic approach, when solving multi-objective mathematical programs, if in step \( p \) we encounter multiple solutions, we should continue with step \( p + 1 \); otherwise, if step \( p \) has a single solution, the subsequent steps repeat, producing the same solution. Thus, it is the multiplicity of the solutions at each stage that feeds the subsequent stage. In a two-stage lexicographic mathematical program where the first objective is the maximization of performance and the second objective is the minimization of buffer size, the maximum performance is on the flat part of the curve; however, there would be no guarantee as to where (i.e. at what buffer level) the observed solution has turned up, but we know that any point on the flat part of the curve produces the same level of performance. In the second stage of the optimization, buffer size is minimized subject to maintaining the same level of performance. In other words, we are moving on the flat portion of the curve (constant output) until we obtain the lowest buffer level. This point is the left point of the flat part of the curve, the point where the flat part starts. This point will be explained in the numerical example section.

3.3. Constraints

The constraints of the model include the following:

\[ d_{i,j} \geq N_j p_{i,j} + s_{i,j}, \quad \forall i \& j \]  
\[ N_{-k} \leq b_{i-1} [d_{i-1,j-k} - s_{i-1,j-k}] \]
\[ \times \left[ - \sum_{r=0}^{k} d_{i-r,j} - \sum_{r=0}^{k} d_{i-1,j-r} - s_{i-1,j-r} + p_{i,j} + s_{i,j-k} \right]^{-1} \]

for \( i = 2, \ldots, m; j = 1, 2 \ldots n; \) and \( k = 0, 1, 2, \ldots, n - 1, \)

\[ N_{j} \geq b_{i}[d_{i+1,j} - s_{i+1,j}] \]
\[ \times \left[ \sum_{r=0}^{k} d_{i+1,j-r} - \sum_{r=0}^{k} d_{i+1,j-r} - p_{i,j-k} + s_{i+1,j-k} \right]^{-1} \]

for \( i = 1, 2, \ldots, m - 1; \quad j = 1, 2 \ldots, n; \) and \( k = 0, 1, 2, \ldots, n - 1, \)

\[ C_{1,j} = d_{1,j}, \quad \forall j \geq 1 \]
\[ C_{i,j} \geq C_{i,j-1} + d_{i,j}, \quad \forall i \geq 1, j \geq 2, \]
Two points are important to mention before the general discussion of the constraints. First, if the maximum makespan, $M_n$, is not specified, then constraints (4)–(8) may be eliminated. Second, if a minimum amount of a product type $N_j$ is mandatory, a simple constraint to that effect can be added to the system.

Because the duration $d_{i,j}$ (duration for which a workstation is occupied by a product), accounts for set up, processing, blocking and starvation, it must be at least as much as the setup plus processing time. This is expressed in constraint (1).

To avoid filling up or exhausting buffers, Johri (1987) suggested two sets of constraints, which are called the input-side and the output-side constraints. These constraints have been modified and used in our proposed model in the form of constraints (2) and (3). Constraint (2) states that the total amount of any product type produced at any workstation must not exceed the sum of the amount produced in the upstream workstation and the buffer between the two workstations; otherwise, starvation would occur. Constraint (3) states that the total amount produced at any workstation must be at least equal to the sum of the amount produced in the downstream workstation and the buffer between the two workstations; otherwise, blocking would result. Please refer to Johri (1987) for a detailed explanation of how this mechanism works. The nonlinear components of our model reside in constraints (2) and (3) for cases when $N_j$ is considered as a variable rather than being a given parameter.

Constraints (4)–(8) relate to the makespan. Constraint (4) initializes the completion time of the first batch on the first workstation. Constraint (5) states that the completion time of batch $j$, on a workstation $i$, is at least equal to the completion time of the previous batch plus the time allocated to this batch. Constraints (6) and (7) refer to our approach in passing on items between workstation and buffer, i.e., item transfer. As each item is transferred to the next workstation individually, a batch of product type $j$ can start on workstation $i$ as soon as at least one item of product $j$ is transferred to workstation $i$. Constraint (6) states that the completion time of batch $j$, on a workstation $i$, is at least equal to the completion time of the batch on the previous workstation plus the time needed in workstation $i$. Constraint (7) states that product $j$ cannot be completed on workstation $i$ until all items of product $j$ are processed on workstation $i$, which is started only after the previous product, product ($j-1$) is completed on workstation ($i-1$), workstation ($i-1$) is set up for product $j$ and at least one item of product $j$ is completed on workstation ($i-1$) and transferred to workstation $i$. We mentioned earlier that if the maximum makespan $M_n$ is not specified, then constraints (4)–(8) may be eliminated.
Constraint (9) states that the total time a workstation is occupied cannot be more than the manufacturing capacity, $T$. The remaining constraints are the non-negativity constraints on the decision variables.

4. Numerical example

The model as presented is a mixed integer nonlinear program, MINLP, which is an NP-hard class of problems. Because of this complexity, it is unlikely that a polynomial-time algorithm exists for these problems (unless $P = NP$). In reference to Bussieck and Vigerske (2010), using commercial solvers such as Lingo may generate a global optimal solution.

The numerical example is intended to demonstrate how the model can be used to respond to various questions about the behavior of the production system. Some typical questions are listed below:

- Given manufacturing capacity $T$, how does the $\Sigma N_j$ change with the changes in buffer size?
- Given demand $N_j \forall j$, how does the max$_j \Sigma d_{ij}$ change with the changes in buffer size?
- Given demand $N_j \forall j$, how does the $\Sigma d_{ij}$ change with the changes in buffer size?
- Given demand $N_j \forall j$, how does the makespan, $C_{m,n}$, change with the changes in buffer size?

Aside from the investigation of the behavior of the system, a decision-maker may want to optimize one or more of the objectives listed in Section 3.2 or consider other unlisted objectives.

The working of the model is demonstrated by means of an example from the automotive industry with the aim to optimize the performance of the cylinder head machining process for three different engine sizes.

The machining process is performed on seven workstations that are equipped with CNC machines. The three products are:

- Product 1: 2.4 L four-cylinder head
- Product 2: 3.0 L six-cylinder head
- Product 3: 3.6 L six-cylinder head

Table 1 presents the input for processing times and setup times for every product in every workstation.

<table>
<thead>
<tr>
<th>Workstation</th>
<th>Product</th>
<th>Processing times (s)</th>
<th>Setup times (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>1. Exhaust and top face</td>
<td></td>
<td>300</td>
<td>325</td>
</tr>
<tr>
<td>2. Bottom and top face</td>
<td></td>
<td>310</td>
<td>340</td>
</tr>
<tr>
<td>3. Intake, top and bottom face</td>
<td></td>
<td>395</td>
<td>425</td>
</tr>
<tr>
<td>4. Bottom and top face</td>
<td></td>
<td>400</td>
<td>430</td>
</tr>
<tr>
<td>5. Top and bottom face</td>
<td></td>
<td>300</td>
<td>310</td>
</tr>
<tr>
<td>6. Top face and tappet hole</td>
<td></td>
<td>345</td>
<td>377</td>
</tr>
<tr>
<td>7. Each face finish and cam</td>
<td></td>
<td>325</td>
<td>397</td>
</tr>
</tbody>
</table>

Table 1: Input data, processing and setup times for the case study.
For the above example, an analysis of the behavior of the production system is conducted by considering a single objective, and then a multiple-objective optimization is carried out.

First, an investigation of the impact of buffer size on the throughput is carried out. The graph in Fig. 3 shows the optimal values of throughput for various buffer sizes. The results in Fig. 3 are for a given capacity, $T = 144,000 \text{s} or 40 \text{ h} (1 \text{ week of operation})$, where the other input data are those given above in Table 1. To produce this curve, for every one of the ten points on the curve, the value of $S_{bi}$ is set to a fixed quantity and the throughput ($\Sigma N_j$) is maximized.

As expected, the graph reaches a plateau at higher buffer sizes, and this occurs at buffer size 26. Any increase beyond this size does not contribute to increasing the throughput. The graph appears jagged for two reasons. First, we have not attempted every possible buffer size. The second reason is the fact that both buffer size and throughput are integer variables and one does not expect to produce a smooth curve using integer values. For example, a buffer size of 18 is needed to produce 409 units. To increase the throughput to 410 units, we need a buffer of size 26. Hence, it is evident that any buffer size between 19 and 25 will only result in a throughput of 409 units.

Having access to a profile as demonstrated in Fig. 3 is helpful to decision-makers to assess the impact of increasing investment in buffer spaces. As indicated earlier in section 3.2, a small reduction in the throughput value close to the plateau does result in a large reduction in the buffer size. For example, if the decision-maker is prepared to accept a throughput of 408 instead of 410, the required buffer size declines from 26 to 14. However, this chart does not demonstrate how such buffer size is to be allocated to the six buffers on the production line.

The next issue to demonstrate is the use of the model to determine the decision variables to achieve an optimal level for a multiple-objective scenario. Our objectives, in the order of preemptive priorities, include maximization of the throughput; minimization of the makespan; and minimization of the buffer size. Hence, the complete objective for a lexicographic goal programming model would be

\[
\text{Maximize } Z = p_1 (\text{throughput}) - p_2 (\text{makespan}) - p_3 (\text{buffer size}).
\]

The lexicographic goal program will be solved in three phases. In phase 1, the throughput is maximized. In phase 2, the maximum throughput is added as a new constraint and the makespan
is minimized. In phase 3, the optimal values of the throughput and makespan are added as additional constraints and the total buffer size is minimized. The results of the first phase indicate that the optimal throughput is 410 units. The second phase indicates that the optimal makespan is 146,100 s. And finally, the third phase produces an optimal buffer size of 26. It should be mentioned that all nonlinear integer programs are solved using LINGO-9 software. Table 2 contains the optimal values of batch sizes and buffer allocations. If the decision-maker finds that the product mix (176, 95, 139) is not satisfactory, additional constraints may be introduced to induce certain relations such as forcing values that are closer to each other. For example, if one wants to consider the relative size of each batch, there would be many possibilities. One of them can be a constraint to force the three batches to be equal in size. The introduction of new constraint in some cases may not affect the feasible region, but in other cases, results in reducing the size of the feasible region, hence potentially deteriorating the optimal solution.

It is coincident that the optimal buffer size is 26, the same value found for optimizing a single objective of throughput as observed on the graph in Fig. 3. Quite often, when the second objective (here the makespan) is introduced, the buffer size ends up increasing. A total buffer size of 26 can be distributed among the six buffers in a large number of ways. Some are not so efficient and cannot support a throughput of 410. The arrangement presented in Table 2 supports a throughput of 410, and the best possible makespan for such throughput. More results are presented in Tables 3 and 4.

<table>
<thead>
<tr>
<th>Table 2</th>
<th>Optimal batch sizes and buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_1$</td>
<td>176</td>
</tr>
<tr>
<td>$N_2$</td>
<td>95</td>
</tr>
<tr>
<td>$N_3$</td>
<td>139</td>
</tr>
<tr>
<td>$\Sigma N_j$</td>
<td>410</td>
</tr>
<tr>
<td>$b_1$</td>
<td>2</td>
</tr>
<tr>
<td>$b_2$</td>
<td>6</td>
</tr>
<tr>
<td>$b_3$</td>
<td>5</td>
</tr>
<tr>
<td>$b_4$</td>
<td>2</td>
</tr>
<tr>
<td>$b_5$</td>
<td>5</td>
</tr>
<tr>
<td>$b_6$</td>
<td>6</td>
</tr>
<tr>
<td>$\Sigma b_j$</td>
<td>26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 3</th>
<th>Workstation utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 1</td>
<td>Station 2</td>
</tr>
<tr>
<td>Total operating time, $OT$, s</td>
<td>133,441</td>
</tr>
<tr>
<td>Total $d(i)$, s or $\sum d_{ij}$</td>
<td>143,620</td>
</tr>
<tr>
<td>Utilization</td>
<td>0.92667</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 4</th>
<th>Workstation’s blocking/starvation in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station 1</td>
<td>Station 2</td>
</tr>
<tr>
<td>Product 1</td>
<td>5,280</td>
</tr>
<tr>
<td>Product 2</td>
<td>2,116</td>
</tr>
<tr>
<td>Product 3</td>
<td>2,783</td>
</tr>
<tr>
<td>Total</td>
<td>10,179</td>
</tr>
<tr>
<td>All 7 stations</td>
<td>18,929</td>
</tr>
</tbody>
</table>

© 2011 The Authors.
International Transactions in Operational Research © 2011 International Federation of Operational Research Societies
From Table 3, row 1 represents the total operating time a station is processing parts. The utilization per station is obtained by dividing the value shown in row 1 by the manufacturing capacity, $T = 144,000\, s$. The total $d(i)$ is the total time the station is dedicated to processing the product mix including setup and blocking/starvation times; this is given by $\sum_j d_{i,j}$. The difference between the entries in row 2 and row 1 of Table 3 gives the results shown in the last row of Table 4 presented below. This accounts for the total time when a station was blocked and/or starved. Table 4 provides more details about a station’s starvation/blocking while processing every individual product of the mix.

By inspecting the results presented in Tables 3 and 4, it can be concluded that blocking/starvation times are causing the decrease in station utilization; however, such a decline is at most 7.3% in station 1.

To further demonstrate the model’s utilization, we present three other scenarios. Each of these scenarios is explained in Table 5.

**Case 1:** The lexicographic goal programming model would be:

Maximize $Z = p_1 \text{(throughput)} - p_2 \text{(makespan)} - p_3 \text{(buffer size)} - p_4 \left( \max_i \sum_j d_{i,j} \right)$. 

We shall consider the optimal results obtained earlier ($N, C_{m,n}$ and individual buffer values) as constraints and determine the optimal value of the fourth objective. The variables would be the individual $N_j$ values.

The optimal value of the fourth objective function,

\[
\max_i \sum_j d_{i,j} = 143,990 \, s.
\]

In Table 6, it can be noticed that while the total throughput was fixed at 410, the individual lot sizes $N_1, N_2$ and $N_3$ are different from those obtained in Table 2.

We notice from row 2 of Table 7 that there is a reduction of 10 s in the largest time allocation (station 4) compared with the results presented in Table 3. This amount is not significant; in other words, the fourth phase of optimization has not been very productive. We also notice a slight increase in machine utilization for workstations 1, 3, 6 and 7 in the last row of Table 7. Other results of workstation blocking/starvation times are shown in Table 8 below, where the sum of
We solved this problem with a different fourth objective function. This time, the fourth objective was "Minimize total production time \( \Sigma \Sigma d_{i,j} \)." Hence, the overall objective function would be:

\[
\text{Maximize } Z = p_1 (\text{throughput}) - p_2 (\text{makespan}) - p_3 (\text{buffer size}) - p_4 \left( \Sigma \Sigma d_{i,j} \right).
\]

We are reporting only the optimal value of \( \Sigma \Sigma d_{i,j} \), which turned out to be 1,002,606 s compared with 1,006,480 s in the base example. This is a reduction of 1.08 h or 0.4%. Small gains in the fourth phase of the two examples covered under case 1 are not surprising. It is the nature of the lexicographic goal programming that, in successive stages of optimization, the size of the feasible region reduces; hence, the chance of improvements decreases successively.

**Case 2:** Case 2 is similar to the first problem of case 1, with the exception that the third objective (minimize buffer size) is eliminated; hence, the fourth objective becomes the third objective. Hence, the overall objective function would be as follows:

\[
\text{Maximize } Z = p_1 (\text{throughput}) - p_2 (\text{makespan}) - p_3 \left( \max_i \Sigma_j d_{i,j} \right).
\]
As the first two phases of this problem have been solved in our base example, we will adopt the following two constraints: \( N = 410 \) and \( M_n = 146,100 \) and proceed to the third phase of the optimization to minimize \( \max_i \sum_j d_{ij} \). The optimal value of \( \max_i \sum_j d_{ij} \) is obtained to be 143,910 s. The reduction is equal to 90 s compared with the base case (144,000 s for station 4). This is a very insignificant gain. This small gain has been achieved at the expense of a significant increase in the total buffer size (from 26 to 51) as shown in Table 9.

The second problem in case 2 is:

\[
\text{Maximize } Z = p_1 \text{(throughput)} - p_2 \text{(makespan)} - p_3 \left( \sum \sum d_{ij} \right).
\]

In this problem, the third objective is minimizing the total production time \( \sum \sum d_{ij} \).

The optimal value of \( \sum \sum d_{ij} \) is obtained to be 989,305 s. This requires a total buffer size of 66, as reported in Table 10.

A fourth objective to minimize \( \Sigma b_i \) was introduced to create problem 3 in case 2. The overall objective would then be:

\[
\text{Maximize } Z = p_1 \text{(throughput)} - p_2 \text{(makespan)} - p_3 \left( \sum \sum d_{ij} \right) - p_4 \left( \sum b_i \right).
\]

The optimal value of \( \Sigma b_i \) was obtained to be 63, a small reduction from 66. The results are presented in Table 11.

Case 3: The overall objective function of the first problem under case 3 is as follows:

\[
\text{Maximize } Z = p_1 \text{(throughput)} - p_2 \left( \max_i \sum_j d_{ij} \right).
\]

Table 9
Optimal batch sizes and corresponding buffers

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>( N_3 )</th>
<th>( \Sigma N_j )</th>
<th>( b_1 )</th>
<th>( b_2 )</th>
<th>( b_3 )</th>
<th>( b_4 )</th>
<th>( b_5 )</th>
<th>( b_6 )</th>
<th>( \Sigma b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>137</td>
<td>119</td>
<td>154</td>
<td>410</td>
<td>22</td>
<td>7</td>
<td>6</td>
<td>2</td>
<td>6</td>
<td>8</td>
<td>51</td>
</tr>
</tbody>
</table>

Table 10
Optimal batch sizes and buffers

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>( N_3 )</th>
<th>( \Sigma N_j )</th>
<th>( b_1 )</th>
<th>( b_2 )</th>
<th>( b_3 )</th>
<th>( b_4 )</th>
<th>( b_5 )</th>
<th>( b_6 )</th>
<th>( \Sigma b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>140</td>
<td>113</td>
<td>157</td>
<td>410</td>
<td>30</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>66</td>
</tr>
</tbody>
</table>

Table 11
Optimal buffers sizes

<table>
<thead>
<tr>
<th>( N_1 )</th>
<th>( N_2 )</th>
<th>( N_3 )</th>
<th>( \Sigma N_j )</th>
<th>( b_1 )</th>
<th>( b_2 )</th>
<th>( b_3 )</th>
<th>( b_4 )</th>
<th>( b_5 )</th>
<th>( b_6 )</th>
<th>( \Sigma b_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>176</td>
<td>95</td>
<td>139</td>
<td>410</td>
<td>30</td>
<td>6</td>
<td>7</td>
<td>5</td>
<td>7</td>
<td>8</td>
<td>63</td>
</tr>
</tbody>
</table>
Phase 1 of the optimization produces $N = 410$. In the second phase, we obtained the following values: \( \min \left[ \max_i \sum_j d_{ij} \right] = 143,805 \text{ s}, \ C_{m,n} = 147,047 \text{ s} \) and \( \Sigma b_i = 68 \text{ units} \). The rest of the results are presented in Table 12.

The second problem under case 3 is

\[
\text{Maximize } Z = p_1 \left( \text{throughput} \right) - p_2 \left( \sum \sum d_{ij} \right).
\]

Phase 1 of the optimization produces $N = 410$. In the second phase, we obtained the following values: \( \min \Sigma \sum d_{i,j} = 990,905 \text{ s}, \ C_{m,n} = 146,223 \text{ s}, \ N_1 = 100, \ N_2 = 133, \ N_3 = 177 \) and \( \Sigma b_i \) was extremely large = \( 0.5 \times 10^{10} \).

The third problem under case 3 starts with the following constraints: $N = 410$, \( C_{m,n} = 146,100 \) and \( \Sigma \Sigma d_{i,j} = 990,905 \text{ s} \). The minimization of \( \Sigma b_i \) produces an optimal value of 58. The rest of the results are presented in Table 13.

The multitude of the problems presented above demonstrates the utilization of our multi-objective model.

### 5. Conclusions and recommendation for future research

This paper has addressed the problem of production output optimization in a highly automated production line composed of serially arranged machines with buffers in between. A nonlinear mathematical programming model was developed to describe the situation. A set of three objectives (throughput, makespan and buffer size) were considered and the multi-objective model was solved using the lexicographic goal programming method. A numerical example was presented and the effects of buffer size on optimal throughput were demonstrated. The numerical example also discusses how a decision-maker may use the model to gain an insight into the behavior of the production line or use it to optimize the production process. Buffer size can contribute to increasing the output level but this contribution decreases for larger buffer sizes. The implication for production managers is to establish the most economical use of buffer space to increase the total throughput. This would result in better inventory management and better
allocation of buffer capacity to different buffer locations. The model may be used with one or multiple objectives as it was exhibited in the example. It can also be modified by introducing new constraints to yield certain desired relations between the throughputs of different product types.

Several extensions to this research are possible. If the problem is converted to cost minimization or profit maximization, then the parameters would be the relative costs or profits of the individual parts rather than their processing times. A second interesting question is the determination of the break-even value of the available capacity, which would allow the examination of the impact of buffers on processing time. If the manufacturing schedule is subject to market demand for the products, as it usually is, then the optimal behavior would be affected by such market demand variations. As a corollary, the effect of the buffer on the production schedule can also be investigated. The explicit introduction of stochastic factors in the production environment would further complicate the system behavior, but these can be investigated through stochastic programming.

Acknowledgements

The authors acknowledge the constructive comments made on the manuscript by three anonymous referees. They also acknowledge the Natural Sciences and Engineering Research Council of Canada (NSERC) for the financial support received during the tenure of this project.

References


© 2011 The Authors.
International Transactions in Operational Research © 2011 International Federation of Operational Research Societies