AN OVERVIEW OF TRADEOFF CURVES IN MANUFACTURING SYSTEMS DESIGN*

GABRIEL R. BITRAN AND REINALDO MORABITO
Massachusetts Institute of Technology, Sloan School of Management, Cambridge, Massachusetts 02139, USA
Universidade Federal de São Carlos, Dept. Engenharia de Produção, 13565-905 São Carlos, SP, Brazil

In this paper we review the use of tradeoff curves in the design of manufacturing systems that can be modeled as open queueing networks. We focus particularly on the tradeoff between expected work-in-process (or product leadtime) and capacity investment in job shops. We review the algorithms in the literature to derive tradeoff curves and illustrate their application in evaluating the efficiency of the system, in deciding how much capacity to have, how to allocate resources between the reduction of uncertainty and the introduction of new technologies, and how to assess the impact of changes in products throughput and product mix. The methodology is illustrated with an example derived from an actual application in the semiconductor industry.

TRADEOFF CURVE ANALYSIS; MANUFACTURING SYSTEM DESIGN; OPEN QUEUEING NETWORKS; OPTIMIZATION AND PERFORMANCE EVALUATION

1. Introduction

A large portion of products is produced in batches in discrete manufacturing systems. Such systems can often be represented by queueing network models where the nodes correspond to the manufacturing stations, and the arcs, connecting the nodes, indicate the flows between stations. Surveys by Buzacott and Yao (1986), Kouvelis and Tirupati (1991), Bitran and Dasu (1992), Buzacott and Shanthikumar (1993), Hsu, Topiero, and Lin (1993), and Suri, Sanders, and Kamath (1993) provide a broad description of the use of queueing networks to analyze discrete manufacturing systems.

Manufacturing networks of queues are often difficult to manage because they process a variety of products that have different characteristics, share the same resources, and are affected by a range of sources of uncertainty like demand variability, suppliers, and processes reliability. The complexity of such systems can be reduced by a better understanding of the tradeoffs among performance measures and resource allocation at the time the system is designed. Several papers analyzing these tradeoffs can be found in the literature, most of them modeling manufacturing systems as closed queueing networks. For example, Shanthikumar and Yao (1987, 1988) explored certain properties of the throughput function and studied the problem of allocating servers in a flexible manufacturing system (FMS) to maximize throughput. Stecke and Solberg (1985) and Dallery and Stecke (1990) con-
sidered the issue of both server and workload allocation. Schweitzer and Seidmann (1991) applied optimization techniques to mean value analysis to determine optimal machine processing rates for FMS. Askin and Krisht (1994) explored the application of optimization procedures for the purpose of analyzing the tradeoff between expected work-in-process (WIP) and processing rates in a FMS.

Other authors analyzed manufacturing systems that can be modeled as open networks of queues. Calabrese (1992) studied workload allocation optimization problems in job shops and FMS represented as Jackson networks and focused on the tradeoffs between throughput and average congestion. Bitran and Tirupati (1989a, 1989b) discussed the notions of tradeoff curves and developed optimization procedures to study the relationship between WIP and capacity in generalized Jackson networks (i.e., networks with general interarrival and service time distributions). These procedures were refined in Bitran and Sarkar (1994). Boxma, Kan, and Van Vliet (1990), Frenk, Labbe, Van Vliet, and Zhang (1994), and Sundarraj, Sundararaghavan, and Fox (1994) studied problems concerning the optimal allocation of servers to workstations in Jackson networks and the tradeoffs between WIP and capacity costs. Van Vliet and Rinnooy Kan (1991) extended this analysis to generalized Jackson networks.

The purpose of this paper is to review the literature on the derivation and use of tradeoff curves for manufacturing networks of queues. We concentrate on the tradeoffs between WIP and capacity in job shops that can be modeled as generalized Jackson networks (the curves are easily adapted to reflect leadtimes instead of WIP). A central issue addressed in the paper is that of selecting among various designs for the network, more specifically, how should financial resources be distributed to allocate capacity at the various stations. We review algorithms for selecting the optimal allocation and deriving the curves. Tradeoff curves are optimal frontier curves. Given a WIP cost, the curve contains the point with the minimum cost of resources. Figure 1 depicts an example of such a curve. A firm competing on the basis of low WIP (or low leadtime) may choose, for example, point A. On the other hand, if it desires to compete on the basis of low investment in capacity (resources), its choice may be, for example, point B with higher WIP. As we will see in Section 3.3, for a given throughput, product mix, and product arrival and processing uncertainty, the curve of Figure 1 corresponds to an efficient frontier for the job shop under consideration.

This paper is organized as follows. In Section 2 we briefly present the decomposition method to evaluate performance measures. In Sections 3.1 and 3.2 we review algorithms for the problem of minimizing WIP without adding resources to the system and the problem

![Figure 1. Point O is the Current State of the System, Points on Curve Have Minimum Resource Cost Given WIP Cost.](image-url)
of minimizing resources without increasing system WIP, respectively. In Section 3.3 we discuss how to utilize these algorithms to generate tradeoff curves. We extend the analysis for the case of additional constraints on capacity when it is not fully transferable from station to station. In Section 4 we review the use of tradeoff curves to analyze the effects of reducing network uncertainty (Section 4.1), changing the throughput (Section 4.2), and the product mix (Section 4.3) of the network. In Sections 5 and 6 we consider the cases where we have a finite set of discrete alternatives for capacity change and where we cannot approximate each station as a single machine. Finally, in Section 7 we present the conclusions of this paper.

To illustrate the presentation of this topic, we chose an example derived from a real situation of a job-shop system with 10 product classes and 13 stations. We generated different tradeoff curves to analyze this network as presented in Sections 3–5.

2. Performance Measures Evaluation: A Summary

In this section we briefly present the decomposition method to evaluate performance measures of open queueing networks. For more details on the method, the readers are referred to Shanthikumar and Buzacott (1981), Whitt (1983), Bitran and Tirupati (1988), Kouvelis and Tirupati (1991), Suri, Sanders, and Kamath (1993), Whitt (1994), and Bitran and Morabito (1996). This procedure utilizes only two parameters (the mean and the squared coefficient of variation (scv)) to approximate the effects of probability distributions of product interarrival and service times at the stations. All measures are computed assuming that the network is in steady-state.

The decomposition method involves basically three steps. In Step 1 the original network is decomposed into a set of independent stations, in Step 2 performance measures are evaluated at each station analyzed as a single queueing system, and in Step 3 performance measures are evaluated for the whole network by combining the results obtained in Step 2. Let’s consider an open queueing network representing a discrete manufacturing system with multiple product classes, deterministic routings, and stations with infinite waiting spaces. In addition to the assumption of steady-state, the approximation described above assumes that the product arrival and departure processes at a station are renewal processes.

Consider that we have the following input data: \( \{ \lambda'_k, ca'_k, n_{kl}, k = 1, \ldots, r; l = 1, \ldots, n_k; \mu_j, cs_j, j = 1, \ldots, n \} \), where each parameter is defined as

\[ \begin{align*}
\lambda'_k &= \text{mean external arrival rate of product class } k, \\
c{a'}_k &= \text{scv of external interarrival times of product class } k, \\
n_{kl} &= \text{station that produces the } l\text{-th operation in the routing of class } k, \\
r &= \text{number of product classes in the network} \\
n_k &= \text{number of operations in the routing of class } k \\
\mu_j &= \text{mean processing rate (or capacity) at station } j \\
 cs_j &= \text{scv of processing times at station } j \\
n &= \text{number of stations in the network.}
\end{align*} \]

Tables 1 and 2 present these parameters for a job-shop network example with 10 product classes and 13 stations. This is derived from a real example of a semi-conductor factory and was analyzed in Bitran and Tirupati (1989b). For simplicity, we consider each station \( j \) as a single machine with mean processing rate \( \mu_j \). In Section 6 we make some comments of how to extend this discussion and consider each station as a set of machines. In addition to the parameters above, Table 2 also presents the parameters \( v_j, a_j, b_j \) to be defined in Sections 2.2 and 3. Note that the network throughput (or mean production rate), defined as \( \Sigma_{k=1,\ldots,r} \lambda'_k \), is equal to 10 products per time unit (see Table 1).
2.1 Step 1

In Step 1 the original queueing network is decomposed into a set of independent single queues corresponding to the stations. The goal is to rewrite the initial parameters \{ \lambda_i, ca_i, n_{kl}, k = 1, \ldots, r; l = 1, \ldots, n_k; \mu_j, cs_j, j = 1, \ldots, n \} as \{ \lambda_j, ca_j, \mu_j, cs_j, j = 1, \ldots, n \}, where \lambda_j and \mu_j denote, respectively, the mean product arrival rate and the scv of product interarrival times at station j. Note that in this way we are aggregating all products into a single class, called the aggregate class, and analyzing its flow through the network. Each parameter \lambda_j is easily obtained by adding the mean external arrival rate of all product classes that visit station j times the number of visits defined in their routings, that is,

$$\lambda_j = \sum_{k=1}^{r} \lambda_i \sum_{l=1}^{n_k} 1\{n_{kl} = j\}$$

where \(1\{n_{kl} = j\} = \begin{cases} 1 & \text{if } n_{kl} = j \\ 0 & \text{otherwise.} \end{cases} \)

The parameters \(ca_j\) are approximately evaluated by solving a linear system as a function of \(ca_j, cd_j\) and \(cd_{k,l}\), defined as follows (Bitran and Tirupati 1988):

$$ca_j = \sum_{k=1}^{r} \lambda_i \sum_{l=1}^{n_k} cd_{k,l-1} 1\{n_{kl} = j\}$$

(1)

$$cd_j = \left(\frac{\lambda_j}{\mu_j}\right)^2 cs_j + \left[1 - \left(\frac{\lambda_j}{\mu_j}\right)^2\right] ca_j$$

(2)

$$cd_{k,l} = \frac{\lambda_i}{\lambda_j} cd_{n_{kl}} + \left(1 - \frac{\lambda_i}{\lambda_j}\right) \frac{\lambda_i}{\lambda_j} + \left(1 - \frac{\lambda_i}{\lambda_j}\right)^2 cd_{k,l-1}$$

(3)

where \(cd_{k,0} = ca_k\), and \(cd_j\) (or \(cd_{n_{kl}}\) for \(n_{kl} = j\)) and \(cd_{k,l}\) are implicit variables in the system (1)–(3). Equation (1) relates to the merging process of product arrivals at each station \(j\), Equation (2) to the departure process of the aggregate class from station \(j\), and Equation (3) to the splitting process of product departures, reflecting the interference among classes. Table 3 presents the parameters \(\lambda_i\) and \(ca_i\) computed for the network example of Tables 1 and 2 (the remaining columns \(\rho_j, L_j, W_j\) and \(F_j\) of Table 3 are defined in Sections 2.2 and 3).

2.2 Steps 2 and 3

In Step 2 we evaluate performance measures of the aggregate class at each station, such as mean capacity utilization, mean product waiting time, mean number of products in

<table>
<thead>
<tr>
<th>Class (k)</th>
<th>(\lambda_i)</th>
<th>(ca_i)</th>
<th>(n_{kl})</th>
<th>(n_k)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1.0</td>
<td>0.500</td>
<td>1, 2, 4, 2, 9, 10, 11</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>1.0</td>
<td>0.500</td>
<td>1, 2, 5, 2, 8, 9, 10, 11</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>1.0</td>
<td>0.333</td>
<td>1, 2, 6, 4, 2, 9, 12, 11</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
<td>0.333</td>
<td>1, 2, 7, 4, 2, 9, 10, 11</td>
<td>8</td>
</tr>
<tr>
<td>5</td>
<td>1.0</td>
<td>0.333</td>
<td>1, 2, 4, 12, 2, 9, 2, 13</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>1.0</td>
<td>0.333</td>
<td>1, 2, 5, 12, 2, 9, 7, 13</td>
<td>8</td>
</tr>
<tr>
<td>7</td>
<td>1.0</td>
<td>0.250</td>
<td>1, 2, 6, 12, 2, 8, 2, 13</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>1.0</td>
<td>1.000</td>
<td>1, 2, 3, 7, 4, 12, 2, 8, 6, 9, 2, 13</td>
<td>12</td>
</tr>
<tr>
<td>9</td>
<td>1.0</td>
<td>1.000</td>
<td>1, 2, 3, 5, 4, 6, 12, 2, 8, 2, 10, 6, 13</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>1.0</td>
<td>0.333</td>
<td>1, 2, 3, 6, 2, 4, 12, 7, 2, 9, 11, 5, 13</td>
<td>13</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>10.0</strong></td>
<td></td>
<td></td>
<td><strong>93</strong></td>
</tr>
</tbody>
</table>
TABLE 2
Input Data for the Stations of the Network Example

<table>
<thead>
<tr>
<th>Station j</th>
<th>$\mu_j$</th>
<th>$cs_j$</th>
<th>$v_j$</th>
<th>$a_j$</th>
<th>$b_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.004</td>
<td>0.500</td>
<td>100</td>
<td>5.68</td>
<td>-51.69</td>
</tr>
<tr>
<td>2</td>
<td>27.778</td>
<td>0.250</td>
<td>1,612</td>
<td>2.59</td>
<td>-50.40</td>
</tr>
<tr>
<td>3</td>
<td>3.160</td>
<td>0.333</td>
<td>733</td>
<td>74.77</td>
<td>-49.73</td>
</tr>
<tr>
<td>4</td>
<td>10.000</td>
<td>0.500</td>
<td>1,052</td>
<td>6.93</td>
<td>-48.54</td>
</tr>
<tr>
<td>5</td>
<td>5.631</td>
<td>0.333</td>
<td>912</td>
<td>12.62</td>
<td>-47.93</td>
</tr>
<tr>
<td>6</td>
<td>9.225</td>
<td>0.250</td>
<td>1,683</td>
<td>7.51</td>
<td>-48.54</td>
</tr>
<tr>
<td>7</td>
<td>5.999</td>
<td>1.000</td>
<td>1,662</td>
<td>11.11</td>
<td>-46.67</td>
</tr>
<tr>
<td>8</td>
<td>4.500</td>
<td>0.333</td>
<td>1,812</td>
<td>27.66</td>
<td>-87.11</td>
</tr>
<tr>
<td>9</td>
<td>10.000</td>
<td>0.333</td>
<td>1,730</td>
<td>7.47</td>
<td>-52.27</td>
</tr>
<tr>
<td>10</td>
<td>5.711</td>
<td>0.333</td>
<td>1,600</td>
<td>15.34</td>
<td>-61.30</td>
</tr>
<tr>
<td>11</td>
<td>5.441</td>
<td>0.333</td>
<td>1,882</td>
<td>27.03</td>
<td>-102.94</td>
</tr>
<tr>
<td>12</td>
<td>7.440</td>
<td>0.500</td>
<td>1,486</td>
<td>13.01</td>
<td>-67.74</td>
</tr>
<tr>
<td>13</td>
<td>7.502</td>
<td>0.500</td>
<td>3,250</td>
<td>14.22</td>
<td>-74.67</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>115.391</td>
</tr>
</tbody>
</table>

queue and in processing, etc. These measures are estimated for each station $j$ by substituting the four parameters \{$\lambda_j$, $ca_j$, $\mu_j$, $cs_j$\} from Step 1 into the appropriate formulae. For example, the mean capacity utilization at station $j$, defined as $\rho_j = \lambda_j/\mu_j$, can be easily calculated with $\lambda_j$ and $\mu_j$. We can also calculate the WIP (in monetary value) at station $j$, defined as:

$$W_j = v_j L_j (\lambda_j, ca_j, \mu_j, cs_j),$$

where $L_j(\lambda_j, ca_j, \mu_j, cs_j)$ is the mean number of products in queue and in processing at station $j$, approximated by a slight modification of Kraemer and Lagenbach-Belz's formula (Whitt, 1983):

$$L_j(\lambda_j, ca_j, \mu_j, cs_j) = \frac{\left(\frac{\lambda_j}{\mu_j}\right)^2 (ca_j + cs_j) g(\lambda_j, ca_j, \mu_j, cs_j)}{2 \left(1 - \frac{\lambda_j}{\mu_j}\right)} + \frac{\lambda_j}{\mu_j}$$

TABLE 3
Parameters and Performance Measures of the Network Example

<table>
<thead>
<tr>
<th>Station j</th>
<th>$\lambda_j$</th>
<th>$ca_j$</th>
<th>$\mu_j$</th>
<th>$L_j$</th>
<th>$W_j$</th>
<th>$F_j$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10.0</td>
<td>0.492</td>
<td>0.769</td>
<td>1.974</td>
<td>197.377</td>
<td>288.325</td>
</tr>
<tr>
<td>2</td>
<td>25.0</td>
<td>0.601</td>
<td>0.900</td>
<td>4.298</td>
<td>6,929.058</td>
<td>598.457</td>
</tr>
<tr>
<td>3</td>
<td>3.0</td>
<td>0.760</td>
<td>0.949</td>
<td>10.694</td>
<td>7,838.522</td>
<td>223.823</td>
</tr>
<tr>
<td>4</td>
<td>7.0</td>
<td>0.608</td>
<td>0.700</td>
<td>1.569</td>
<td>1,650.825</td>
<td>207.700</td>
</tr>
<tr>
<td>5</td>
<td>4.0</td>
<td>0.613</td>
<td>0.710</td>
<td>1.500</td>
<td>1,367.930</td>
<td>120.993</td>
</tr>
<tr>
<td>6</td>
<td>6.0</td>
<td>0.583</td>
<td>0.650</td>
<td>1.118</td>
<td>1,881.390</td>
<td>191.323</td>
</tr>
<tr>
<td>7</td>
<td>4.0</td>
<td>0.619</td>
<td>0.667</td>
<td>1.715</td>
<td>2,850.717</td>
<td>119.836</td>
</tr>
<tr>
<td>8</td>
<td>4.6</td>
<td>0.665</td>
<td>0.889</td>
<td>4.403</td>
<td>7,979.090</td>
<td>168.193</td>
</tr>
<tr>
<td>9</td>
<td>8.0</td>
<td>0.642</td>
<td>0.800</td>
<td>2.327</td>
<td>4,025.622</td>
<td>224.300</td>
</tr>
<tr>
<td>10</td>
<td>4.0</td>
<td>0.662</td>
<td>0.700</td>
<td>1.489</td>
<td>2,382.308</td>
<td>150.240</td>
</tr>
<tr>
<td>11</td>
<td>5.0</td>
<td>0.684</td>
<td>0.919</td>
<td>6.194</td>
<td>11,656.346</td>
<td>240.055</td>
</tr>
<tr>
<td>12</td>
<td>7.0</td>
<td>0.614</td>
<td>0.941</td>
<td>9.226</td>
<td>13,709.098</td>
<td>216.225</td>
</tr>
<tr>
<td>13</td>
<td>6.0</td>
<td>0.677</td>
<td>0.800</td>
<td>2.653</td>
<td>8,620.970</td>
<td>240.110</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td></td>
<td></td>
<td>49.160</td>
<td>71,089.253</td>
<td>2,988.689</td>
</tr>
</tbody>
</table>
TRADEOFF CURVES IN MANUFACTURING SYSTEM DESIGN

\[ g(\lambda_j, c\lambda_j, \mu_j, c\sigma_j) = \begin{cases} 
\exp \left\{ \frac{-2 \left( 1 - \frac{\lambda_j}{\mu_j} \right)(1 - c\lambda_j)^2}{3 \frac{\lambda_j}{\mu_j} (c\lambda_j + c\sigma_j)} \right\} & \text{if } c\lambda_j < 1 \\
1 & \text{if } c\lambda_j \geq 1 
\end{cases} \]

and \( \nu_j \) is a unit monetary value of an arbitrary product at station \( j \). Each value \( \nu_j \) is estimated using practical experience or as a weighted average proportional to the expected arrival rate and expected waiting time of each class (the expected waiting time may be computed approximately by a procedure given in Albin (1986)). Obviously, if \( \nu_j = 1 \), then \( W_j \) corresponds to the mean number of units of products at station \( j \). Table 2 presents the values of \( \nu_j \) and Table 3, the computed values of \( \rho_j, L_j \) (5) and \( W_j \) (4) at each station \( j \) of the network example.

Finally, in Step 3 we want to evaluate performance measures for the whole network, such as mean product leadtimes and network WIP. These measures can be estimated by combining the results obtained for each station in Step 2. For example, adding the WIP of all stations we obtain the network WIP defined as \( W = \sum_{j=1,...,n} W_j \), with value 71,089 (Table 3). These performance measures refer to the aggregate class; as we decompose these results we can obtain performance measures for each original class.

It is worth noting that the decomposition method is an approximate procedure that was known in the literature to perform well. For instance, Bitran and Tirupati (1988) suggested that the estimates given by (1)–(3) are typically within 5% to 10% of the simulation figures (in their experiments, the average error of the mean number of products with networks similar to the one of Tables 1 and 2 was 5.95%). This accuracy is acceptable in many practical situations. The potential for applications motivated the development of various software packages based on decomposition methods, such as QNA (Segal and Whitt 1989), Q-LOTS (Karmarkar, Kekre, and Freeman 1985), Manuplan (Suri, Diehl, and Dean 1986), Operations Planner (Jackman and Johnson 1993), and MPX (Suri, Diehl, De Treville, and Tomsicek 1995).

3. Tradeoff Curve Generation

In this section we discuss how to generate tradeoff curves between capacity resources and the WIP of the network (this methodology can be easily extended to other performance measures, such as product leadtimes). For convenience, the resources are measured by a cost function of capacity defined below. In Section 3.1 we discuss how to minimize network WIP without adding resources to the system and in Section 3.2, how to minimize the resources without changing network WIP. As we will see, the solutions to these problems correspond to points on the tradeoff curve. Finally, in Section 3.3 we discuss how to use these results to generate the remaining points on the curve.

To present tradeoff curves for the network example of the preceding section, we measure the system resources by the cost of capacity acquisition (or capacity investment). This cost is a function of the capacity at each station \( j \), \( \mu_j \). An example of such a cost function is:

\[ F_j(\mu_j) = a_j \mu_j^2 + b_j \mu_j + c_j, \tag{6} \]

where \( a_j, b_j \), and \( c_j \) are known coefficients (the use of quadratic functions for capacity cost is common in the literature; see, e.g., Hax and Candea 1984). We are assuming that it is possible to add capacity to station \( j \) with amounts small enough to consider \( \mu_j \) as a continuous variable. In Section 5 we analyze the more general case where capacity changes are limited to a finite set of discrete alternatives.
Table 2 presents the values of $u_j$ and $b_j$ for each station of the network example (for simplicity, we assume that $c_j = 0$). Adding the capacity cost of all stations, we obtain the network resources defined as: $F = \sum_{j=1}^{n} F_j(\mu_j)$. Table 3 presents these values utilizing the data of Table 2. Note that the WIP value 71,089 and the resource value 2,989 define the point $O$ depicted in Figure 1. For convenience, we assume that the network capacity is homogeneous and interchangeable among stations. An example is a trained labor force that can be transferred from one station to others. The algorithms presented below can also be applied when the capacity of a station is not transferable to all other stations. At the end of Section 3.3 we discuss this more general case.

3.1 Efficient Redistribution of Resources

Let’s assume that the system is at point $O$ (71,089, 2,989). Considering the data of Tables 1 and 2, we can formulate the following question: Is it possible to reduce the network WIP to below 71,089 without adding resources to the network? In other words, is it possible to redistribute the resources of 2,989 (by interchanging capacity between stations) such that the network WIP is reduced? And if this reduction is possible, what is the redistribution that leads to the minimum network WIP? The optimal resource redistribution problem can be solved by the following exact iterative algorithm proposed in Bitran and Sarkar (1994).

**Algorithm 1**

**Step 0:** Given the initial parameters \( \lambda_k', c_{ak}', n_k, k = 1, \ldots, r; \mu_j^0, c_s, j = 1, \ldots, n \), apply the decomposition method described in Section 2.1 to obtain the parameters \( \lambda_j, c_{aj}, \mu_j^0, c_s, j = 1, \ldots, n \), where \( c_{aj} \) and \( \mu_j^0 \) denote, respectively, the initial scv of interarrival times and the initial capacity at station $j$. Define $F_T = \sum_{j=1}^{n} F_j(\mu_j^0)$ and set the iteration index $p = 1$.

**Step 1:** For each iteration $p$, use the scv $c_{aj}^{p-1}$, $j = 1, \ldots, n$ to solve the following nonlinear optimization problem as a function of $\mu_j$:

\[
\begin{align*}
\min \ W &= \sum_{j=1}^{n} W_j(\mu_j) \\
\text{s.t.:} \ &\sum_{j=1}^{n} F_j(\mu_j) = F_T \\
\text{with:} \ &\mu_j > \lambda_j \quad \text{for } j = 1, \ldots, n
\end{align*}
\]

where $W_j(\mu_j)$ and $F_j(\mu_j)$ are defined according to (4) and (6). Let $\mu_j^p, j = 1, \ldots, n$, denote the optimal solution of problem (7) - (9) using $c_{aj}^{p-1}$.

**Step 2:** Apply the decomposition method to the parameters \( \lambda_k', c_{ak}', n_k, k = 1, \ldots, r; \mu_j^0, c_s, j = 1, \ldots, n \) in order to obtain the parameters \( \lambda_j, c_{aj}^{p}, \mu_j^p, c_s, j = 1, \ldots, n \). Stop if $c_{aj}^{p-1}$ and $c_{aj}^{p}$ were sufficiently close; otherwise, make $p = p + 1$ and go back to Step 1.

In Algorithm 1 we assume that as the capacity $\mu_j$ varies, the squared mean $E(S_j)^2$ and the variance $V(S_j)$ of processing times at station $j$ vary in the same proportion, and, hence, $c_s$ remains nearly constant. We also assume that the scv $c_{aj}$ is independent of capacity changes at the stations during each iteration. Under these assumptions it can be shown that $W_j(\mu_j)$ is a convex function of $\mu_j$ (Bitran and Tirupati 1989a). Therefore, the problem (7) - (9) is convex in the variables $\mu_j, j = 1, \ldots, n$, and can be solved by any efficient convex programming technique described, for example, in Bazaraa, Sherali, and Shetty (1993). Bitran and Sarkar (1994) have shown that algorithm 1 converges to an optimal solution under conditions usually found in practice.

Applying Algorithm 1 to the network example, we obtain the point $A$ (49254, 2989) depicted in Figure 1. The algorithm converges after two iterations for an accuracy of 0.001
in the \( c_{aj} \) values. Table 4 presents the final values of \( c_{aj}, \mu_j, \rho_j, L_j, W_j, \) and \( F_j \) for each station. Note that the values of \( c_{aj} \) in Tables 3 (point O) and 4 (point A) are almost the same in spite of the capacity changes at stations.

Point A indicates that we can substantially reduce the network WIP (from 71,089 to 49,254) without changing the resources (2,989). This reduction is obtained by appropriately redistributing the resources among stations (compare Tables 3 and 4). The throughput is also maintained, equal to 10 units of product per time unit (see Table 1). As we change the system state from point O to point A, we must sell capacity of Stations 1, 4, 5, 6, 7, 9, and 10 in order to buy capacity for Stations 2, 3, 8, 11, 12, and 13. Although the network capacity at point O, 115.4, is different from the network capacity at point A, 112.2 (compare Tables 2 and 4), their cost are exactly the same: 2,989. The WIP increases a little at Stations 1, 4, 5, 6, 7, 9, and 10, but decreases substantially at Stations 3, 8, 11, and 12.

Note that if we make \( v_j = l, j = 1, \ldots, n, \) then \( W_j = L_j \) in (4), and Algorithm 1 redistributes the available resources \( F_T \) such that the mean number of products in queue and in processing over the network is minimized. Furthermore, if we make \( a_j = 0, b_j = 1, c_j = 0, j = 1, \ldots, n, \) then \( F_j = \mu_j \) in (6) and \( F_T = \Sigma_{j=1}^{n} \mu_j^0 \), and Algorithm 1 now redistributes capacity instead of resources (capacity investment). In this case we say that we are balancing the system (recall that we have assumed that all capacity is homogeneous and interchangeable).

### 3.2 Efficient Redistribution of WIP

Let’s assume that the system is again at point O (71,089, 2,989). Considering the data of Tables 1 and 2, we can formulate the following (second) question: Is it possible to reduce the network resources to under 2,989 without changing the network WIP of 71,089? In other words, is it possible to redistribute the WIP of 71,089 (by interchanging capacity among stations) such that the necessary network resources are reduced? And if this reduction is possible, what is the redistribution that leads to the minimum value of network resources?

The optimal WIP redistribution problem can be solved by an exact iterative algorithm, similar to Algorithm 1. The same assumptions with respect to the convexity of \( W_j \) in (4) and the convergence of the procedure are assumed for Algorithm 2, described below.

### Table 4

<table>
<thead>
<tr>
<th>Station ( j )</th>
<th>( c_{aj} )</th>
<th>( \mu_j )</th>
<th>( \rho_j )</th>
<th>( L_j )</th>
<th>( W_j )</th>
<th>( F_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.492</td>
<td>10.604</td>
<td>0.943</td>
<td>8.609</td>
<td>860.861</td>
<td>90.541</td>
</tr>
<tr>
<td>2</td>
<td>0.602</td>
<td>28.041</td>
<td>0.892</td>
<td>3.966</td>
<td>6,392.554</td>
<td>623.270</td>
</tr>
<tr>
<td>3</td>
<td>0.761</td>
<td>3.421</td>
<td>0.877</td>
<td>4.279</td>
<td>3,136.301</td>
<td>309.222</td>
</tr>
<tr>
<td>4</td>
<td>0.610</td>
<td>8.712</td>
<td>0.804</td>
<td>2.587</td>
<td>2,721.365</td>
<td>103.173</td>
</tr>
<tr>
<td>5</td>
<td>0.621</td>
<td>5.081</td>
<td>0.787</td>
<td>2.141</td>
<td>1,952.440</td>
<td>73.096</td>
</tr>
<tr>
<td>6</td>
<td>0.589</td>
<td>7.818</td>
<td>0.767</td>
<td>1.787</td>
<td>3,007.797</td>
<td>79.567</td>
</tr>
<tr>
<td>7</td>
<td>0.624</td>
<td>5.828</td>
<td>0.686</td>
<td>1.874</td>
<td>3,115.223</td>
<td>105.356</td>
</tr>
<tr>
<td>8</td>
<td>0.665</td>
<td>4.999</td>
<td>0.800</td>
<td>2.369</td>
<td>4,292.979</td>
<td>255.741</td>
</tr>
<tr>
<td>9</td>
<td>0.643</td>
<td>9.918</td>
<td>0.807</td>
<td>2.415</td>
<td>4,178.749</td>
<td>216.376</td>
</tr>
<tr>
<td>10</td>
<td>0.666</td>
<td>5.296</td>
<td>0.755</td>
<td>1.891</td>
<td>3,026.341</td>
<td>105.638</td>
</tr>
<tr>
<td>11</td>
<td>0.682</td>
<td>6.050</td>
<td>0.826</td>
<td>2.795</td>
<td>5,260.484</td>
<td>366.620</td>
</tr>
<tr>
<td>12</td>
<td>0.611</td>
<td>8.403</td>
<td>0.833</td>
<td>3.101</td>
<td>4,607.640</td>
<td>349.405</td>
</tr>
<tr>
<td>13</td>
<td>0.678</td>
<td>7.987</td>
<td>0.751</td>
<td>2.062</td>
<td>6,701.743</td>
<td>310.684</td>
</tr>
<tr>
<td>Total</td>
<td>112.158</td>
<td>39.876</td>
<td>49,254.477</td>
<td>2,988.689</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Algorithm 2
Step 0: Given the initial parameters \{\lambda_l, ca^l_j, n_l, k = 1, \ldots, r; l = 1, \ldots, n_l; \mu_j^0, cs_j, j = 1, \ldots, n\}, apply the decomposition method described in Section 2.1 to obtain the parameters \{\lambda_j, ca_j^0, \mu_j^0, cs_j, j = 1, \ldots, n\}, where \(ca_j^0\) and \(\mu_j^0\) denote, respectively, the initial scv of interarrival times and the initial capacity at station \(j\). Define \(W_r = \sum_{j=1}^{n} W_j(\mu_j^0)\) and set the iteration index \(p = 1\).

Step 1: For each iteration \(p\), use the scv \(ca_j^{p-1}, j = 1, \ldots, n\) to solve the following convex program in the variables \(\mu_j^p\):

\[
\begin{align*}
\min F &= \sum_{j=1}^{n} F_j(\mu_j) \\
\text{s.t.:} &\quad \sum_{j=1}^{n} W_j(\mu_j) = W_T \\
&\quad \mu_j > \lambda_j \quad \text{for} \; j = 1, \ldots, n,
\end{align*}
\]

where \(W_j(\mu_j)\) and \(F_j(\mu_j)\) are defined according to Expressions (4) and (6). Let \(\mu_j^p, j = 1, \ldots, n\), denote the optimal solution of problem (10) - (12) using \(ca_j^{p-1}\).

Step 2: Apply the decomposition method to the parameters \{\lambda_l, ca^l_j, n_l, k = 1, \ldots, r; l = 1, \ldots, n_l; \mu_j^p, cs_j, j = 1, \ldots, n\} in order to obtain the parameters \{\lambda_j, ca_j^p, \mu_j^p, cs_j, j = 1, \ldots, n\}. Stop if \(ca_j^{p-1}\) and \(ca_j^p\) were sufficiently close; otherwise, make \(p = p + 1\) and go back to Step 1.

Applying Algorithm 2 to the network example, we obtain point \(B\) \((71,089, 2,278)\) depicted in Figure 1, which parameters and performance measures appear in Table 5. The algorithm converges after two iterations for an accuracy of 0.001 in the \(ca\) values. Note that, similarly to point A, the values of \(ca_j\) at point \(B\) (Tables 5) are very close to the values at point \(O\) (Table 3) in spite of the capacity changes at the stations.

Point \(B\) indicates that we can reduce the network resources from 2,989 to 2,278 without changing the network wip of 71,089 (compare Tables 3 and 5). Similarly to the efficient redistribution of resources, the efficient redistribution of wip does not imply a process, technology, or throughput change. As we move the system state from point \(O\) to point \(B\), we transfer wip from Stations 3, 8, 11, and 12 to other stations. This transfer is obtained by appropriately interchanging capacity between stations such that the network wip of

<table>
<thead>
<tr>
<th>Station (j)</th>
<th>(ca_j)</th>
<th>(\mu_j)</th>
<th>(\rho_j)</th>
<th>(L_j)</th>
<th>(W_j)</th>
<th>(F_j)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.492</td>
<td>10.390</td>
<td>0.962</td>
<td>13.114</td>
<td>1,311.434</td>
<td>76.112</td>
</tr>
<tr>
<td>2</td>
<td>0.598</td>
<td>26.978</td>
<td>0.927</td>
<td>5.841</td>
<td>9,415.943</td>
<td>525.354</td>
</tr>
<tr>
<td>3</td>
<td>0.760</td>
<td>3.275</td>
<td>0.916</td>
<td>6.355</td>
<td>4,658.410</td>
<td>260.353</td>
</tr>
<tr>
<td>4</td>
<td>0.607</td>
<td>8.143</td>
<td>0.860</td>
<td>3.730</td>
<td>3,923.519</td>
<td>64.327</td>
</tr>
<tr>
<td>5</td>
<td>0.616</td>
<td>4.720</td>
<td>0.847</td>
<td>3.039</td>
<td>2,772.018</td>
<td>46.459</td>
</tr>
<tr>
<td>6</td>
<td>0.581</td>
<td>7.215</td>
<td>0.832</td>
<td>2.489</td>
<td>4,189.145</td>
<td>40.719</td>
</tr>
<tr>
<td>7</td>
<td>0.617</td>
<td>5.255</td>
<td>0.761</td>
<td>2.686</td>
<td>4,463.872</td>
<td>61.536</td>
</tr>
<tr>
<td>8</td>
<td>0.657</td>
<td>4.660</td>
<td>0.858</td>
<td>3.402</td>
<td>6,163.572</td>
<td>194.739</td>
</tr>
<tr>
<td>9</td>
<td>0.638</td>
<td>9.270</td>
<td>0.863</td>
<td>3.465</td>
<td>5,994.499</td>
<td>157.403</td>
</tr>
<tr>
<td>10</td>
<td>0.657</td>
<td>4.868</td>
<td>0.822</td>
<td>2.664</td>
<td>4,262.773</td>
<td>65.111</td>
</tr>
<tr>
<td>11</td>
<td>0.672</td>
<td>5.690</td>
<td>0.879</td>
<td>4.047</td>
<td>7,616.165</td>
<td>289.389</td>
</tr>
<tr>
<td>12</td>
<td>0.604</td>
<td>7.923</td>
<td>0.884</td>
<td>4.537</td>
<td>6,741.802</td>
<td>279.960</td>
</tr>
<tr>
<td>13</td>
<td>0.668</td>
<td>7.330</td>
<td>0.819</td>
<td>2.946</td>
<td>9,576.101</td>
<td>216.651</td>
</tr>
<tr>
<td>Total</td>
<td>105.717</td>
<td>58.315</td>
<td>71,089.253</td>
<td>2,278.113</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
71,089 is maintained. Note that the resources increase a little at Stations 3, 8, 11, and 12, but decrease by more than half of their initial values at stations 1, 4, 6, and 10.

Similarly to Algorithm 1, if we make \( U_j = 1, j = 1, \ldots, n \), then \( W_j = \sum_{i=1}^{n} L_j(\mu_j^i) \) since \( W_j = L_j \) in (4). Thus, Algorithm 2 now redistributes the mean number of products \( W_j \) in the network such that the required resources \( F \) is minimized. Furthermore, if we make \( a_j = 0, b_j = 1, c_j = 0, j = 1, \ldots, n \), then \( F_j = \mu_j \) in (6), and Algorithm 2 determines the minimum capacity to maintain the mean number of products \( W_j \) in the network.

3.3 Efficient Frontier

The Algorithms 1 and 2 move the system to a point on the tradeoff curve depicted in Figure 1 (recall that points A and B belong to the curve). Algorithm 1 moves the system from point O to point A by efficiently redistributing the resources of point O, while Algorithm 2 moves the system from point O to point B by efficiently redistributing the WIP of point O. The remaining points on the curve can also be obtained applying Algorithm 1 (Algorithm 2) for arbitrary values of network resources \( F_T \) (network WIP \( W_T \)) in Step 1 of the algorithm. In particular, the curve of Figure 1 was traced applying Algorithm 2 to the network WIP values \( W_T = 40,000, 50,000, \ldots, 90,000 \) indicated in the figure (see the corresponding dots in the figure that originate the curve).

Alternatively, this curve can also be generated with less computational effort using a heuristic algorithm proposed in Bitran and Tirupati (1989a). The algorithm assumes that the system is at a point on the curve and employs a simple and intuitive greedy heuristic to find the remaining points. This procedure is illustrated in the following example. Consider that we want to add 100 labor hours of capacity to the stations of a network. For simplicity, assume that the cost of adding 1 hour to any station is constant, let’s say $1 (and hence, allocating 100 hours is equivalent to allocating $100 to the network). The question is, how should we distribute this extra capacity to the stations such that the network WIP is minimized?

Given that we can add capacity to the stations in small quantities, let’s partition these 100 hours in sufficiently small increments and add them, one after another, according to the following greedy rule. The next increment is added to the station that results in the largest reduction of the network WIP, and so on, until all increments have been added to the network. The smaller the increments, the more accurate is the solution generated by this procedure. The complete description of the heuristic algorithm can be found in Bitran and Tirupati (1989a), and Bitran and Morabito (1996).

The tradeoff curve of Figure 1 defines an efficient frontier, that is, the minimum resource value necessary to produce each WIP or, equivalently, the minimum WIP produced by each value of resources. Let’s take for example point A (49,254, 2,989) and consider that, according to the competitive strategy, the system should operate with a WIP less than or equal to 40,000. What is the minimum resource requirement to reduce the WIP from 49,254 to 40,000? As we travel through the points on the curve to the left of point A, we find point C (40,000, 3,609). Hence, the system needs an additional capacity investment of 620 (3,609–2,989).

HETEROGENEOUS AND NONINTERCHANGEABLE CAPACITY. For convenience, we have assumed so far that the network capacity is homogeneous and totally interchangeable between stations. Algorithms 1 and 2 can also be applied when part of the capacity \( \mu_j \) at station \( j \) is not interchangeable; in this case it is enough to impose a lower bound on variable \( \mu_j \) in Step 1 of the algorithms. To do this, we just add to problems (7)-(9) and (10)-(12) the constraint \( \mu_j \geq \mu_j^i, j = 1, \ldots, n \), where \( \mu_j^i \) corresponds to the noninterchangeable capacity at station \( j \). Other type of constraints can also be added to reflect the restrictions in transferring resources among stations.
The more general case where capacity need not be homogeneous nor interchangeable involves additional considerations. For example, if we are allowed only to add capacity to the stations, we should include in problems (7)–(9) and (10)–(12) the constraint \( \mu_j \geq \mu_j^0 \), \( j = 1, \ldots, n \), where \( \mu_j^0 \) is the initial capacity at station \( j \). In this case Algorithms 1 and 2 can be applied for arbitrary values of \( F_T \) and \( W_T \) in Step 1, such that \( F_T = \sum_{j=1}^{n} F_j(\mu_j^0) \) and \( W_T = \sum_{j=1}^{n} W_j(\mu_j^0) \). An example occurs in the design of a new manufacturing network, or the redesign of an existing one, when we can not sell (i.e., remove) capacity of one station in order to get money to buy (i.e., add) capacity to another station. The case where we are also allowed to sell capacity of the stations may require modifications to the algorithms as discussed below.

When capacity is not transferable among stations, we may have to sell capacity from one to acquire capacity for another. In many practical instances the sale of capacity leads to a financial loss. In such cases Equation (8) does not hold and can be replaced by constraint (14) below, where \( d_j \) is a positive number less than or equal to 1, reflecting the financial loss in the transaction. Algorithm 1 can be adapted to solve this situation by replacing the convex program (7)–(9) of Step 1 with (similarly in Algorithm 2):

\[
\min W = \sum_{j=1}^{n} W_j(\mu_j) \tag{13}
\]

s.t. \( \sum_{j=1}^{n} d_j \max \{0, F_j(\mu_j^0) - F_j(\mu_j)\} = \sum_{j=1}^{n} \max \{0, F_j(\mu_j) - F_j(\mu_j^0)\} \) \tag{14}

with \( \mu_j > \lambda_j \) \( \text{for} \ j = 1, \ldots, n. \) \tag{15}

\( W_j(\mu_j) \) and \( F_j(\mu_j) \) are defined according to (4) and (6) (note that \( F_j \) must be a non-decreasing function of \( \mu_j \)). Let us define \( y_j = \max \{0, F_j(\mu_j^0) - F_j(\mu_j)\} \) and \( z_j = \max \{0, F_j(\mu_j) - F_j(\mu_j^0)\} \), and replace the max functions by disjunctive constraints (Nemhauser and Wolsey 1988). We can rewrite the previous problem as:

\[
\min W = \sum_{j=1}^{n} W_j(\mu_j) \tag{16}
\]

s.t. \( \sum_{j=1}^{n} d_j y_j = \sum_{j=1}^{n} z_j \) \tag{17}

\[
y_j \geq F_j(\mu_j^0) - F_j(\mu_j) \quad \text{for all} j \tag{18}
\]

\[
y_j \leq M (1 - p_j) \quad \text{for all} j \tag{19}
\]

\[
z_j \geq F_j(\mu_j) - F_j(\mu_j^0) \quad \text{for all} j \tag{20}
\]

\[
z_j \leq M (1 - q_j) \quad \text{for all} j \tag{21}
\]

\[
p_j + q_j = 1 \quad \text{for all} j \tag{22}
\]

with \( \mu_j > \lambda_j, y_j \geq 0, z_j \geq 0, p_j, q_j \in \{0, 1\}, \ j = 1, \ldots, n, \tag{23} \)

where \( M \) is a sufficiently large positive number, and \( p_j \) and \( q_j \) are control variables. Note that if \( d_j = 1, j = 1, \ldots, n \), then problems (16)–(25) and (7)–(9) are equivalent (see the Appendix) and Algorithm 1 can be applied without any modification. The concepts of points \( A, B \), and the efficient frontier remain valid.
In the next section we assume that the efficient redistribution of resources (or WIP) has been done and the system state is at a certain point on the curve of Figure 1, let’s say at point A (49,254, 2,989). In the preceding discussion we have shown that, starting from a point on the curve, we can reduce WIP by adding capacity to the network. In the sequel we discuss other alternatives to reduce WIP, such as uncertainty reduction. The next tradeoff curves to be presented were also generated with Algorithms 1 and 2.

4. Changing the Variability Parameters, Throughput, and Product Mix

The tradeoff curve of Figure 1 was generated with the data of Tables 1 and 2, where the variability parameters (i.e., the \( c_{a_k} \) for all product classes and the \( c_{S_j} \) for all stations), the throughput, and the product mix remained fixed. We varied the capacity \( \mu_j \) at each station and consequently, we varied the resources, the WIP, and the mean utilization at each station. In this section we analyze what happens to that tradeoff curve as we change the variability parameters, the throughput, and the product mix of the network.

4.1 Changing the Variability Parameters

There are many endogenous and exogenous factors that contribute to uncertainty. Examples of endogenous factors are poorly trained operators, machine breakdowns, maintenance failures, shortages, etc. These sources of uncertainty may be controlled, for example, by investing in labor training and process improvement. In general, we have more control over endogenous factors than exogenous factors. But it is also often possible to manage the uncertainty of exogenous factors, for example, by working closer with suppliers or reducing the total product cycle time (including design and production).

As we reduce the variability parameters \( c_{a_k} \), \( k = 1, \ldots, r \), and \( c_{S_j} \), \( j = 1, \ldots, n \), we expect a flattening effect of the tradeoff curves between network resources and WIP. In this way we obtain lower WIP levels without additional capacity investments. An immediate question is, under what conditions does uncertainty reduction produce better performance (e.g., lower WIP levels) than simply investing in capacity expansion?

Figure 2 presents the tradeoff curve of Figure 1 (curve 1) next to three other curves generated with smaller values of \( c_{a_k} \) and \( c_{S_j} \). In the first (curve 2) we reduce by half all \( c_{a_k} \) values of Table 1, in the second (curve 3) we reduce by half all \( c_{S_j} \) values of Table 2, and in the third (curve 4) we reduce by half all \( c_{a_k} \) and \( c_{S_j} \) values.

![Figure 2. Changing the Variability Parameters: Curve 1 (ca_k, cs_j), Curve 2 (ca_k/2, cs_j), Curve 3 (ca_k, cs_j/2), and Curve 4 (ca_k/2, cs_j/2).](image-url)
Starting from point A (49,254, 2,989), the points B1 (45,584, 2,989), B2 (40,645, 2,989) and B3 (36,948, 2,989) can be obtained with Algorithm 1, and the points C1 (49,254, 2,803), C2 (49,254, 2,537), and C3 (49,254, 2,351) with Algorithm 2. Consider that the system is originally at point A and let's take, for example, the curve 4. Define V as the required investment to reduce by half all variability parameters. As we invest V, we move the system state from point A to point B3 and, hence, we reduce the WIP to 36,948. This WIP level could also be attained by investing 918 (i.e., 3,907 − 2,989) in additional network capacity, instead of variability reduction, to reach point A3 (36,948, 3,907). The value 918 becomes an upper bound on the investment V. Note that with the curves of Figure 2 at hand, we can now measure the tradeoff between investing in capacity versus investing in variability reduction.

The reductions of the scv csj of stations produce a larger reduction than that of the scv ca;j of product classes (compare curves 2 and 3). This can be explained in part by Equation (2) of the scv of interdeparture times at station j, cdj. Note in that equation that, for high utilization at station j, the csj contribution to cdj is larger than that of ca;j. Nevertheless, we expect the inverse for low utilization: this can be observed for WIP values less than 10,000, where curve 2 crosses curve 3 (this is not shown in Figure 2), and the effect of reducing ca;j becomes larger than that of reducing csj.

TECHNOLOGY SUBSTITUTION. As we have seen, utilizing the tradeoff curves of Figure 2, we can assess the tradeoff between adding capacity and investing in uncertainty reduction, without changing either the technology, the throughput, or the product mix of the network. Now let's assume that we have an alternative technology that allows us to produce the same mix of products at the same throughput rate. Figure 3 depicts its hypothetical curve (curve 5) together with the curves of the current technology (curve 1) and the current technology with uncertainty reduction (curve 4). These two last ones correspond, respectively, to the curves 1 and 4 of Figure 2. Note that now we have a new tradeoff analysis: the tradeoff between buying this substitute technology versus investing in uncertainty reduction in the current system.

4.2 Changing the Throughput

The tradeoff curves also help the analysis of throughput changing in the network. Figure 4 presents curve 1 of Figure 1, together with two other curves generated by varying the original network throughput, equal to 10 products per time unit (Table 1). In the first
2°mO 25ccxl 3ocm 35cOo 4ccm 4xXx 5cmo 5m 6ocoO

FIGURE 4. Changing the Throughput: Curve 1 (10 Products/Hour), Curve 2 (9 Products/Hour), and Curve 3 (11 Products/Hour).

(curve 2), we reduce by 10% the mean external arrival rates of all product classes in the network (so that the network throughput becomes nine products per time unit), and in the second (curve 3), we increase them by 10% (11 products per time unit).

Starting at point A (49,254, 2,989), we obtain points B1 (32,079, 2,989) and B2 (98,107, 2,989) (the latter does not appear in figure 4) with Algorithm 1, and points C1 (49,254, 1,798) and C2 (49,254, 4,378) with Algorithm 2. Consider again that the system is at point A and take, for example, curve 3. Note that it is unlikely that the system will survive the 10% growth of the throughput without additional resources (point B2). However, even a 50% increase of the current resources is not sufficient to maintain the same WIP level of point A (point C2).

4.3 Changing the Product Mix

The effects of changes in the product mix, such as removing old products, modifying the proportion among products, and including new products, can also be analyzed with tradeoff curves. Figure 5 presents curve 1 of Figure 1, together with three other curves

FIGURE 5. Changing the Product Mix: Curve 1 (Original Curve), Curve 2 (Class 1 Deleted), Curve 3 (Class 1 Duplicated), and Curve 4 (Class 11 Added).
generated by modifying the product mix. In the first (curve 2) we eliminate product class 1 (i.e., $\lambda_1 = 0$), in the second (curve 3) we double the mean arrival rate of product class 1 (i.e., $\lambda_1 = 2$), and in the third (curve 4) we introduce a new product class (class 11) with the same arrival process of class 1 (i.e., $\lambda'_{11} = 1$ and $\alpha'_{11} = 0.5$) but with a very different routing: the sequence of stations 13, 1, 11, 3, 9, 5, and 7.

Starting at point A (49,254, 2,989), we obtain points B1 (36,607, 2,989), B2 (77,768, 2,989), and B3 (103,211, 2,989) (points B2 and B3 do not appear in Figure 5) with Algorithm 1, and points C1 (49,254, 2,184), C2 (49,254, 3,973) and C3 (49,254, 4,435) with Algorithm 2. Note that if we eliminate product class 1, we reduce the mean utilization of the original network, and the effect on network WIP corresponds to the horizontal distance between points A and B1. On the other hand, if we double product class 1 or introduce product class 11, we raise the mean capacity utilization, yielding to a substantial increase in the network WIP, as shown by points B2 and B3. Similar results were found in Section 4.2 as we increased by 10% the throughput of the network (compare Figures 4 and 5).

5. Discrete Alternatives for Capacity Changes

In Sections 3 and 4 we have assumed that capacity can be added or removed from station $j$ by amounts small enough to consider the total capacity at the station, $\mu_j$, as a continuous variable. This is not always valid. In this section we briefly analyze the more general case where capacity changes at the station are limited to a finite set of discrete alternatives. In a previous paper, Bitran and Tirupati (1989b) presented a heuristic algorithm that considers the capacity $\mu_j$ as a discrete variable. The algorithm presented in this section, called Algorithm 3, can be seen as an extension of that algorithm where the variability parameters are updated at each iteration.

Consider that instead of choosing any value for $\mu_j$, we are limited to a finite set of $n_j$ discrete alternatives for station $j$. This set is described by the vector $\{\mu_{j1}, \mu_{j2}, \ldots, \mu_{jn}\}$, where $\mu_{ji}$ denotes the capacity at station $j$ under alternative $i$ and satisfies $\mu_{ji} > \lambda_j$ for all $i$. Table 6 presents a set with five possible capacity alternatives for each station of the network example. Note that the first alternative corresponds to point $O$ of Figure 1.

<table>
<thead>
<tr>
<th>Station $j$</th>
<th>Alternative 1</th>
<th>Alternative 2</th>
<th>Alternative 3</th>
<th>Alternative 4</th>
<th>Alternative 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>13.004</td>
<td>10.5</td>
<td>11.0</td>
<td>14.0</td>
<td>15.0</td>
</tr>
<tr>
<td>2</td>
<td>27.778</td>
<td>26.0</td>
<td>27.0</td>
<td>28.0</td>
<td>30.0</td>
</tr>
<tr>
<td>3</td>
<td>3.160</td>
<td>3.5</td>
<td>3.5</td>
<td>4.0</td>
<td>4.5</td>
</tr>
<tr>
<td>4</td>
<td>10.000</td>
<td>7.5</td>
<td>8.0</td>
<td>9.0</td>
<td>11.0</td>
</tr>
<tr>
<td>5</td>
<td>5.631</td>
<td>4.5</td>
<td>4.7</td>
<td>5.0</td>
<td>6.0</td>
</tr>
<tr>
<td>6</td>
<td>9.225</td>
<td>6.5</td>
<td>7.0</td>
<td>9.0</td>
<td>12.0</td>
</tr>
<tr>
<td>7</td>
<td>5.999</td>
<td>5.0</td>
<td>5.5</td>
<td>6.0</td>
<td>6.5</td>
</tr>
<tr>
<td>8</td>
<td>4.500</td>
<td>4.5</td>
<td>5.0</td>
<td>5.5</td>
<td>6.0</td>
</tr>
<tr>
<td>9</td>
<td>10.000</td>
<td>8.5</td>
<td>9.0</td>
<td>10.0</td>
<td>11.0</td>
</tr>
<tr>
<td>10</td>
<td>5.711</td>
<td>4.5</td>
<td>4.7</td>
<td>5.0</td>
<td>6.0</td>
</tr>
<tr>
<td>11</td>
<td>5.441</td>
<td>5.3</td>
<td>5.6</td>
<td>5.9</td>
<td>6.0</td>
</tr>
<tr>
<td>12</td>
<td>7.440</td>
<td>7.5</td>
<td>8.0</td>
<td>8.5</td>
<td>9.0</td>
</tr>
<tr>
<td>13</td>
<td>7.502</td>
<td>6.5</td>
<td>7.0</td>
<td>7.5</td>
<td>8.0</td>
</tr>
<tr>
<td>Total</td>
<td>115.391</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

TABLE 6
Five Discrete Alternatives for Capacity Changes at Each Station
Each alternative \( i \) corresponds to the resource requirement \( F_{ji}(\mu_{ji}) \) at station \( j \), defined similarly to expression (6) as

\[
F_{ji}(\mu_{ji}) = a_j \mu_{ji}^2 + b_j \mu_{ji} + c_j.
\]  

(26)

Note that for each alternative we can calculate the corresponding \( F_{ji}(\mu_{ji}) \) in (26). Furthermore, after choosing alternatives for all stations, we may apply Step 1 of the decomposition method (Section 2.1) to obtain the parameters \( \{ \lambda_j, c_{aj}, \mu_j, c_{sj}, j = 1, \ldots, n \} \). Let's assume that we chose alternative \( i \) at station \( j \) (i.e., \( \mu_j = \mu_{ji} \)); so, station \( j \) is described by the four parameters \( \{ \lambda_j, c_{aj}, \mu_{ji}, c_{sj} \} \). Now we can calculate the WIP at station \( j \) under alternative \( i \), \( W_{ji} \), defined similarly to expression (4) as

\[
W_{ji} = u_j L_{ji}(\lambda_j, c_{aj}, \mu_{ji}, c_{sj}),
\]  

(27)

where, as before, \( u_j \) is the unit monetary value of an arbitrary product at station \( j \), and \( L_{ji}(\lambda_j, c_{aj}, \mu_{ji}, c_{sj}) \) is the mean number of products (in queue and in processing) at station \( j \) under alternative \( i \). \( L_{ji} \) can be computed as \( L_{ji} \) by expression (5). Note that, once we have chosen an alternative for each station, we can calculate each \( W_{ji} \) in (27).

Define \( y_{ji} \) as a decision variable at station \( j \) such that

\[
y_{ji} = \begin{cases} 
1 & \text{if alternative } i \text{ is chosen at station } j \\
0 & \text{otherwise} 
\end{cases}
\]

and \( \sum_{i=1}^{n_j} y_{ji} = 1 \). Hence, we are limited to one and only one choice of capacity at each station. As we choose \( y_{ji} = 1 \), we decide to allocate the capacity \( \mu_{ji} \) at station \( j \). Note that we can define the capacity at station \( j \) as \( \mu_j = \sum_{i=1}^{n_j} \mu_{ji} y_{ji} \) and so, all expressions of Section 2 can be rewritten as a function of \( \mu_{ji} \) and \( y_{ji} \).

Consider the optimal WIP redistribution problem described in Section 3.2 (the analysis of the optimal resource redistribution problem is similar). Algorithm 2 can be adapted to deal with discrete alternatives for capacity changes as described below.

**Algorithm 3** (Algorithm 2 for discrete alternatives)

**Step 0:** Given the initial parameters \( \{ \lambda_j, c_{aj}, n_{i_l}, k = 1, \ldots, r; l = 1, \ldots, n_i; \mu_{i_l}^0, c_{sj}, j = 1, \ldots, n \} \), apply the decomposition method described in Section 2.1 to obtain the parameters \( \{ \lambda_j, c_{aj}^0, \mu_j^0, c_{sj}, j = 1, \ldots, n \} \), where \( c_{aj}^0 \) and \( \mu_j^0 \) denote, respectively, the initial scv of interarrival times and the initial capacity at station \( j \) (e.g., \( \mu_j^0 = \mu_{j1}, j = 1, \ldots, n \)). Define \( W_T \) and set the iteration index \( p = 1 \).

**Step 1:** For each iteration \( p \), use the scv \( c_{aj}^{p-1} \), \( j = 1, \ldots, n \), to compute \( W_{ji} \) in (27) for all \( \mu_{ji} \) and to solve the following integer linear programming problem in the variables \( y_{ji} \):

\[
\min F = \sum_{j=1}^{n} F_{ji} y_{ji}
\]

subject to:

\[
\sum_{j=1}^{n} \sum_{i=1}^{n_j} W_{ji} y_{ji} \leq W_T
\]

\[
\sum_{i=1}^{n_j} y_{ji} = 1 \quad \text{for all } j
\]

(30)

with \( y_{ji} \in \{ 0, 1 \} \) for \( j = 1, \ldots, n; i = 1, \ldots, n_j \).

Let \( y_{ji}^p, j = 1, \ldots, n; i = 1, \ldots, n_j \) denote the optimal solution of program (28)–(31) using \( c_{aj}^{p-1} \). Note that if \( y_{ji}^p = 1 \), then capacity \( \mu_{ji} \) is allocated to station \( j \). Let \( \mu_{ji}^p, j = 1, \ldots, n \) be the capacity allocated to station \( j \).

**Step 2:** Apply the decomposition method to the parameters \( \{ \lambda_j, c_{aj}, n_{i_l}, k = 1, \ldots, r; l = 1, \ldots, n_i; \mu_j^p, c_{sj}, j = 1, \ldots, n \} \) in order to obtain the parameters \( \{ \lambda_j, c_{aj}^p, \mu_j^p, c_{sj}, j = 1, \ldots, n \} \).
\[ j = 1, \ldots, n \}. \text{Stop if } ca_j^{p-1} \text{ and } ca_j^p \text{ were sufficiently close or if } p \text{ overtakes a certain threshold; otherwise, make } p = p + 1 \text{ and go back to Step 1.}

The problem (28)–(31) can be solved by integer linear programming techniques known in the literature; see e.g. Nemhauser and Wolsey (1988). In particular Bitran and Tirupati (1989b) proposed a heuristic procedure, based on the linear program relaxation of the problem, that demands little computational efforts to find good solutions. In this present paper the problem (28)–(31) is optimally solved at each iteration of Algorithm 3 using an exact branch-and-bound based routine.

Applying Algorithm 3 to the network example of the previous sections (with the five available alternatives of Table 6), we obtain point \( D (70,927, 2,359) \) indicated in Figure 1 and presented in Table 7. The algorithm converges after three iterations for an accuracy of 0.001 in the \( ca_j \) values and a maximum relative gap of 0.2% from the optimal solution value. Note that it chose different alternatives for the stations (see the second column of the table). The columns \( ca_{ji}, \mu_{ji}, \rho_{ji}, \) and so on indicate the parameters and performance measures at station \( j \) under alternative \( i \) relative to point \( D \).

The \( ca_j \) values at point \( D \), as well as at point \( B \), are very close to the \( ca_j \) values at point \( O \) (compare Tables 3, 5 and 7). Note also that the wip at point \( D \) can be less than or equal to \( W_T (71,089) \), instead of exactly equal to \( W_T \) as at point \( B \) (compare inequality (29) of Algorithm 3 and equality (11) of Algorithm 2). Given the discrete set of capacities \( \mu_{ji} > \lambda_j \) for all \( j \) and \( i \), it follows that the resource requirement at point \( B, 2278 \) (Figure 1) becomes a lower bound on the resource requirement at point \( D \), equal to 2,359 (recall that point \( B \) is obtained from problem (10)–(12) where the capacities (decision variables) \( \mu_j, j = 1, \ldots, n \), are allowed to be continuous).

Similarly to Algorithm 2, we can also apply Algorithm 3 for different values of \( W_T \) in order to trace the efficient frontier of the problem with discrete alternatives for capacity changes. Naturally this efficient frontier now is not defined as a continuous curve anymore but as a set of discrete points. Our first computational experience with Algorithm 3 suggests that it converges under reasonable tolerances in the accuracy of the final values of \( ca_j \) (these tolerances are set in order to prevent cycling near the optimal solution); the proof of its convergence is a topic for future research.

The three algorithms presented in this paper (Algorithms 1–3) were codified in the modeling language \textit{GAMS (General Algebraic Modeling System, version 2.25)} (Brooke, Kendrick, and Meeraus 1992). To solve at each iteration the linear and convex programs

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|c|}
\hline
Station & Alt. & \( ca_{ji} \) & \( \mu_{ji} \) & \( \rho_{ji} \) & \( L_{ji} \) & \( W_{ji} \) & \( F_{ji} \) \\
\hline
1 & 2 & 0.492 & 10.500 & 0.952 & 10.315 & 1,031,498 & 83,475 \\
2 & 3 & 0.598 & 27.000 & 0.926 & 5.782 & 9,321,135 & 527,310 \\
3 & 2 & 0.760 & 3.500 & 0.857 & 3.652 & 2,676,847 & 337,032 \\
4 & 3 & 0.610 & 8.000 & 0.875 & 4.229 & 4,448,631 & 55,280 \\
5 & 4 & 0.619 & 5.000 & 0.800 & 2.285 & 2,084,087 & 66,850 \\
6 & 3 & 0.584 & 7.000 & 0.857 & 2.954 & 4,971,318 & 28,210 \\
7 & 3 & 0.622 & 5.500 & 0.727 & 2.266 & 3,765,365 & 79,392 \\
8 & 1 & 0.660 & 4.500 & 0.889 & 4.386 & 7,947,184 & 168,120 \\
9 & 3 & 0.637 & 9.000 & 0.889 & 4.300 & 7,438,939 & 134,640 \\
10 & 4 & 0.654 & 5.000 & 0.800 & 2.348 & 3,756,943 & 77,000 \\
11 & 3 & 0.671 & 5.600 & 0.893 & 4.597 & 8,651,750 & 271,197 \\
12 & 3 & 0.606 & 8.000 & 0.875 & 4.216 & 6,265,355 & 290,720 \\
13 & 4 & 0.666 & 7.500 & 0.800 & 2.637 & 8,568,902 & 239,850 \\
\hline
Total & & 106.100 & 53.967 & 70,927.955 & 2,359.077 & \\
\hline
\end{tabular}
\caption{Parameters and Performance Measures of Point D}
\end{table}
of Algorithms 1 and 2, we utilized the solver GAMS/Minos and the linear and integer linear programs of Algorithm 3, the solver GAMS/Osl. The solutions presented in Tables 4, 5, and 7, as well as the tradeoff curves, were obtained in a few minutes (including the generation of detailed reports) using a microcomputer PC-AT-486. The computational performance can be improved with the implementation of mathematical routines that explore the particular characteristics of the convex and integer linear programs involved.

6. Multiple Machines

In Sections 3–5 we defined the capacity at each station as the mean processing rate $\mu_j$. We considered each station $j$ as a single machine, or a set of machines, operators, tools, etc., that can be approximated by a single machine with mean processing rate $\mu_j$. This approximation is not always reasonable. There may be situations where we must describe the capacity at each station as a set of machines, each one with a given mean processing rate.

In the case where we have identical machines at each station (i.e., with the same mean processing rate), algorithms very similar to Algorithms 1–3 can be applied. Such algorithms consider the decision variable at each station as the number of machines, instead of the mean processing rate. Furthermore, performance measures such as the WIP defined in expression (4) must be redefined according to the multi-machine formulas of queueing theory. For more details of these algorithms, see e.g., Boxma, Rinnooy Kan, and Van Vliet (1990), Bitran and Tirupati (1989b), Van Vliet and Rinnooy Kan (1991), Frenk, Labbe, Van Vliet, and Zhang (1994), and Bitran and Morabito (1996). The more general case when we may have distinct machines at the same station involves additional difficulties and is a topic for future research.

7. Conclusions

In this paper we reviewed the use of tradeoff curves in the design of manufacturing systems. We emphasized their role in the analysis of complex discrete systems that can be modeled as open networks of queues and used the decomposition method to evaluate performance measures. The generation of tradeoff curves was described in the context of optimization problems combining the allocation of capacity resources and measures of systems performance, more specifically WIP. To illustrate the theory we presented several tradeoff curves for a manufacturing network example of a semiconductor factory and used them to analyze the effects of uncertainty reduction, throughput variation, and product mix changes.

Tradeoff curves can be effectively used to evaluate strategic options as a function of the resources required to meet specific goals to compete effectively in the market place.¹

¹ The authors thank Luis A. C. Pedrosa and Amit Dhadwal of the Sloan School of Management, M.I.T. and the anonymous referees for their helpful comments and suggestions. This research was partially supported by grant #93/0891-7 from Fundação de Amparo a Pesquisa do Estado de São Paulo, Brazil.

Appendix

In this appendix we show that if $d_j = 1$, $j = 1, \ldots, n$, then constraints (8) and (14) are equivalent. First, let us define $N = \{1, 2, \ldots, n\}$. $N$ can be partitioned into the three disjoint sets:

\[
N_1 = \{ j \in N | F_j(\mu_0) - F_j(\mu_j) > 0 \}
\]

\[
N_2 = \{ j \in N | F_j(\mu_0) - F_j(\mu_j) < 0 \}
\]

\[
N_3 = \{ j \in N | F_j(\mu_0) - F_j(\mu_j) = 0 \}
\]
where $N_1$, $N_2$, and $N_3$ are the sets of stations which had their resources reduced, increased, and unchanged, respectively, after the redistribution. Constraint (14) can be rewritten as

$$\sum_{j \in N_1} d_j [F_j(\mu_j^2) - F_j(\mu_j)] = \sum_{j \in N_2} [F_j(\mu_j) - F_j(\mu_j^0)], \quad (32)$$

where the left-hand side of (32) corresponds to the total revenue of selling resources of stations in $N_1$ and the right-hand side to the total cost of acquiring resources for the stations in $N_2$. Since all stations in $N_3$ had their resources unchanged, $\sum_{j \in N_3} F_j(\mu_j) = \sum_{j \in N_3} F_j(\mu_j^0)$. We can rewrite (32) as

$$\sum_{j \in N_1} d_j F_j(\mu_j) + \sum_{j \in N_2} F_j(\mu_j) + \sum_{j \in N_3} F_j(\mu_j) = \sum_{j \in N_1} d_j F_j(\mu_j^0) + \sum_{j \in N_2} F_j(\mu_j^0) + \sum_{j \in N_3} F_j(\mu_j^0), \quad (33)$$

which reduces to (8) for the particular case where $d_j = 1$ for all $j \in N$.

References


Gabriel R. Bitran has a MSc. and a Ph.D. in Operations Research from the Massachusetts Institute of Technology and a BS and a MSc. in industrial engineering from the Escola Politecnica of The University of Sao Paulo, Brazil. He is currently a professor of management science at M.I.T.’s Sloan School of Management. Professor Bitran is a member of the editorial boards of several journals. His research interests are in the field of operations management in manufacturing in the service industry.

Reinaldo Morabito is associate professor of the Production Engineering Department, Federal University of Sào Carlos, Brazil. He has a Ph.D. in Transportation Engineering and a MSc. in Operations Research from the University of São Paulo, and a BSc. in Civil Engineering from the State University of Campinas, Brazil. He was a visiting scholar at the Sloan School of Management, Massachusetts Institute of Technology, from August 1993 to December 1994. His research interests are in the field of operations management and operations research. He has published articles in cutting and packing problems, queueing networks applied to manufacturing systems, combinatorial optimization, and logistics and transportation planning.