Efficient Methods for Manufacturing System Analysis and Design


Abstract — The goal of the research described here is to develop tools to assist the rapid analysis and design of manufacturing systems. The methods we describe are based on mathematical models of production systems. We combine earlier work on the decomposition method for factory performance prediction and design with the hedging point method for scheduling. We propose an approach that treats design and operation in a unified manner. The models we study take into account, including random failures and repairs of machines, finite buffers, random demand, production lines, assembly and disassembly, imperfect yield, and token-based control policies.

Keywords — manufacturing systems, performance analysis, system design, decomposition, optimization

I. INTRODUCTION

The goal of the research described here is to develop tools to assist the rapid analysis and design of manufacturing systems. The need for such tools is due to a variety of trends:

- **Frequent new product introductions.** Product lifetimes are often short. For that reason, and others, process lifetimes are also short. This leads to frequent building and rebuilding of factories. In addition, more frequent changes of product families and production plans require frequent reconfiguration of existing factories.

- **High capital costs.** Some kinds of factories — especially for semiconductor fabrication — cost billions of dollars.

As a result, there is not enough time, or it is too expensive, to build a factory and then improve it after watching it operate. By the time it has been observed long enough to gain experience, the product or process lifetime may be nearly over; or rebuilding the factory to improve its performance would be prohibitively expensive. Furthermore, factories must be run efficiently, and the effects of short-term disruptions (such as machine failures) must be minimized.

**Needs:** Tools are needed to rapidly predict the performance of a proposed factory design. This will make possible tools for the optimal design of a factory. In addition, methods for optimal real-time management (control) of factories are needed. Finally, Manufacturing Systems Engineering professionals are needed who understand the capabilities and limitations of such tools, and who understand factories as complex systems.

**Analytical methods and simulation.** The methods we describe are based on mathematical models of production systems. Quantitative results are obtained by calculation — the solution of systems of equations and related methods. By contrast, simulation, which is widely used for factory analysis and design, is based on creating a detailed representation of every important individual event that occurs in the production process. Both kinds of methods have important advantages. A great advantage of mathematical methods is that they can be much faster. This speed greatly facilitates design experimentation and optimization. Besides improving the resulting factories, such experimentation can greatly improve the intuition of the designers. In addition, there are no statistical significance issues, as there are in simulation.

Two of the major issues treated in the manufacturing systems research literature are performance analysis and decision and control policies. In this research, we combine earlier work by the author and his colleagues on the decomposition method for factory performance prediction and design with the hedging point method for scheduling. We propose an approach that treats design and operation in a unified manner. The method will eventually include algorithms for selecting buffer (in-process inventory storage) sizes and operational parameters.

**A. Background**

Decomposition is an analytical technique for evaluating performance measures (such as production rate, average buffer levels, and probabilities of blockage and starvation) of queueing networks, in which a large system is split into a large set of small systems. It is a useful approximation technique for systems for which exact analytic methods do not exist, particularly systems with finite buffers. A decomposition method was developed for tandem systems with finite buffers in [1] and improved in [2]. It was extended to assembly/disassembly systems in [3] and [4]. It is reviewed in [5] and discussed
in detail in [6] and [7]. It is rapid and accurate for these systems (although no proof of convergence or bounds on accuracy is known). Recent work has extended these methods to systems with yield losses and rework [8], [9], [10].

Gershwin [11] describes a relationship between certain control policies and assembly/disassembly networks. The original material flow network representation is expanded to include the flow of information. Information appears in the form of tokens or kanbans, and operations are permitted only if an upstream token buffer is not empty, and a downstream buffer is not full. (In effect, every controlled operation is an assembly operation, where a part and a token are assembled.)

However, the methods of [3] and [4] cannot always be applied to evaluate performance. These methods can only be applied to tree-structured or acyclic networks. In most cases, however, the expanded network contains loops (or cycles).

For example, Figure 1 represents the flow of material in a three-machine \( (M_1, M_2, M_3) \), two-buffer \( (B_1, B_2) \) manufacturing system. Machine \( M_i \) can only do an operation if its upstream buffer \( B_{i-1} \) is not empty and its downstream buffer \( B_i \) is not full. \( M_i \) is assumed always to have a non-empty upstream buffer and \( M_i \) always has a non-full downstream buffer.) In Figure 2, an information system is added to control material flow. This adds one more condition to \( M_i \); it operates when and only when buffer \( SB_i \) is not full.

The demand buffers \( DB_i \) are infinite. This means that no order is ever lost. The material buffers \( B_i \) and the surplus buffers \( SB_i \) are finite, and their sizes are control parameters.

Current research is aimed at extending decomposition methods to networks with loops. This paper describes the extension to systems that consist only of the machines and buffers in a single loop. A later paper [12] will describe the extension to general assembly/disassembly networks with multiple loops, such as Figure 2.

A closed-loop production system or loop is a system in which a constant amount of material flows through a single fixed cycle of work stations and storage buffers. This type of system occurs frequently in manufacturing. Manufacturing processes which utilize pallets or fixtures can be viewed as loops since the number of pallets/fixtures that are in the system remains constant. Similarly, control policies such as CONWIP and kanban create conceptual loops by imposing a limitation on the number of parts that can be in the system at any given time. Figure 3 represents a \( K \)-machine loop.

![Fig. 1. Production line](image1)

When a demand (i.e., an order) arrives at \( D \), a demand token is sent to each demand buffer \( (DB_1, DB_2, DB_3) \). When machine \( M_i \) performs an operation, it sends the part to buffer \( B_i \) and it sends a production token to surplus buffer \( SB_i \).

![Fig. 2. Production line with information flow system](image2)

Synchronization machine \( S_i \), which never fails, performs an operation if neither of its upstream buffers are empty. When it does, both buffers lose one token each. Finally, when \( M_i \) does an operation, \( B_{i-1} \) loses one part, \( B_i \) gains a part, and \( SB_i \) gains a token.

![Fig. 3. Illustration of a closed-loop production system](image3)

**B. Problem Statement**

Performance measures such as average production rate and the distribution of in-process inventory cannot be expressed in closed form. Simulation provides accurate results for these quantities, but can be time consuming. Some faster analytical methods have been developed, but they can only be used in a limited class of cases (see Section I-C). The purpose of this paper is to summarize a more versatile analytical method for evaluating these performance measures of closed-loop production systems. Specifically, we are concerned with closed-loop systems where the number of parts in the system is larger than the number of machines and the size of the smallest buffer (see Section III-A.1). Details are presented in [13].

**C. Literature Review**

Compared to open transfer lines, relatively little work has been done on closed-loop production systems with finite buffers and unreliable machines. Onural and
Perrus [14] demonstrated that the production rate of a closed-loop system is a function of the number of parts in the system. In addition, they showed that the throughput versus population curve is symmetric when blocking occurs before service and processing time is exponential. To avoid the complication finite buffers create in closed-loop systems, Akyildiz [15] approximated production rate by reducing the population and evaluating the same system with infinite buffers. Boulochouch, Frein, and Dallery [16] used a closed-loop queuing network with finite capacities to model a closed-loop system with finite buffers. For a more detailed listing of previous work dealing with closed-loop systems, see [17].

The first analytical method for evaluating the performance of closed-loop systems with finite buffers and unreliable machines was proposed by Frein, Comnault, and Dallery in 1996 [18]. This method is an extension of the decomposition method developed by Gershwin [1]. It is important to note that this method does not account for the correlation among numbers of parts in each buffer. As a result, the method is only accurate for large loops.

Maggio [17] presents a new decomposition method which does account for the correlation between population and the probability of blocking and starvation. However, the method is too complex to be practical for loops with more than three machines. Here, we extend Maggio’s results.

II. Approach

The models we study take many of the most important features and phenomena in factories into account, including random failures and repairs of machines, finite buffers, random demand, production lines, assembly and disassembly, imperfect yield, and token-based control policies. At present, we deal only with single-product systems in which parts visit each machine no more than once, but we plan to extend the models to include multiple products and reentrant flow.

System performance measures are evaluated or optimized. The measures include production rate, average inventory, and other closely related quantities (such as utilization).

The mathematical methodologies we use include Markov processes and dynamic programming. Approximations (especially decomposition) are required due to the large state spaces and absence of exact solutions for these systems.

A. Closed-Loop Production Systems

We first describe our model of a manufacturing system. Next, we review the existing techniques for evaluating open production lines and explain why the characteristics of loops make these techniques inadequate. Finally, we propose a transformation and decomposition method designed specifically for closed-loop systems.

A.1 Basic Model

Throughout this analysis, we extend the deterministic processing time model presented in [6] to closed-loop systems. More specifically, we use the version of the model presented by Tolio and Matta [19], which allows machines to fail in more than one mode. This feature is critical to our method and is discussed in detail in Section II-A.6. Processing times for all machines are assumed to be deterministic and identical. In addition, all operational (i.e., not failed) machines start their operations at the same time. For simplicity, we scale the processing time to one time unit. Parts in the machines are ignored, as is travel time between machines. Machine failure times and repair times are geometrically distributed.

$M_i$ refers to Machine $i$. $B_i$ is its downstream buffer and has capacity $N_i$, $B_{i+1}$ is the buffer upstream of $M_i$. A machine is blocked if its downstream buffer is full and starved if its upstream buffer is empty. When $M_i$ is working (operational and neither blocked nor starved) it has a probability $p_{ij}$ of failing in mode $j$ in one time unit. If $M_i$ is down in mode $j$, it is repaired in a given time unit with probability $r_{ij}$. By convention, machine failures and repairs take place at the beginning of time units and changes in buffer levels occur at the end of time units. The population, the fixed total number of parts in the system, is $N_p$.

A.2 Transfer Line Decomposition Techniques

Although it is possible to obtain an analytical solution for a two-machine line directly, the problem becomes intractable for longer lines. However, accurate decomposition methods have been developed for evaluating long transfer lines [6]. These methods decompose a $K$-machine transfer line into $K-1$ two-machine lines or building blocks. In each building block $L(i)$, the buffer $B(i)$ corresponds to $B_i$ in the original transfer line. The upstream machine $M^u(i)$ represents the collective behavior of the line upstream of $B_i$ and the downstream machine $M^d(i)$ represents the behavior downstream.

To an observer sitting in $B(i)$, $M^u(i)$ appears to be down when $M_i$ is either down or starved by some upstream machine. Using Tolio’s terminology, $M^u(i)$ has real failure modes corresponding to those of $M_i$ and virtual failure modes corresponding to each of the upstream machines [19]. Likewise, $M^d(i)$ has real failure modes corresponding to those of $M_{i+1}$ and virtual failure modes corresponding to each of the downstream machines.

This is illustrated in Figure 4, which focuses on the view of the observer in the buffer $B_3$, who believes that he is in the buffer of the two-machine line consisting of $M^u(3)$, $B(3)$, and $M^d(3)$. Failure modes are indicated
in the machines, so Machine 1 fails in modes 1 and 2, Machine 2 fails in mode 3, etc. The real modes, as seen by the observer in Buffer 3 are 4, 5, 6, and 7. The rest are virtual.

![Fig. 4. Tolio decomposition](image)

The goal of the decomposition method is to choose the parameters of $M^p(i)$ and $M^d(i)$ such that the flow of parts through $B(i)$ mimics that through $B_i$. Accomplishing this for all building blocks gives approximate values for average throughput and buffer levels in the original transfer line.

A.3 Special Characteristics of Closed-Loop Systems

In a transfer line, blocking and starvation can propagate throughout the entire system. If the first machine fails, it is possible for all of the downstream machines to become starved. Similarly, if the last machine fails, all upstream machines can become blocked.

This is not the case in loops. Whether or not a machine can be starved or blocked by the failure of another machine depends on the number of parts in the system and the total buffer space between the two machines. For ease of notation, we define all subscripts to be modulo $K$. In particular, we define the set $(i, j)$ as:

$$(i, j) = \begin{cases} (i, i+1 \ldots, j) & \text{if } i < j \\ (i, i+1 \ldots, K, 1 \ldots, j) & \text{if } i > j \end{cases} \quad (1)$$

We define $\Psi(v, w)$ as the total buffer capacity between $M_v$ and $M_w$ in the direction of flow [17]. More formally,

$$\Psi(v, w) = \begin{cases} \sum_{z=1}^{w-1} N_z & \text{if } v \neq w \\ 0 & \text{if } v = w \end{cases} \quad (2)$$

The total buffer space in the line is $\Psi(v, w) + \Psi(w, v)$ (for any $w \neq v$) and the population must satisfy

$$0 \leq N^p \leq \Psi(v, w) + \Psi(w, v).$$

If $N^p < \Psi(v, w)$, then the failure of $M_w$ can never cause $M_v$ to become blocked because there are not enough parts in the system to fill all buffers between $M_v$ and $M_w$ simultaneously. Conversely, if $N^p > \Psi(v, w)$, $M_w$ cannot starve $M_v$.

A.4 Thresholds

The issue of blocking and starvation is more complicated still. In some cases, whether or not a machine can ever be starved or blocked by the failure of a specific other machine depends on the number of parts in an adjacent buffer. This is the concept of thresholds introduced in [17].

Consider the case where 7 parts are traveling through a three-machine loop with buffers of size 5 (see Figure 5). If $M_2$ fails, parts begin to build up in $B_1$ and eventually $M_1$ becomes blocked. However, we know that $M_1$ cannot be blocked if the number of parts in its upstream buffer, $B_3$, remains greater than 2. This would mean that the number of parts in its downstream buffer must be less than 5 since there are only 7 parts in the system. Conversely, we know that if the number of parts in $B_3$ remains less than 2 then the number of parts in $B_2$ must be greater than 0 and $M_2$ cannot become starved. Therefore, we say that $B_3$ has a threshold of 2.

![Fig. 5. Example of a Loop with Thresholds](image)

In general, we define the threshold $l_j(i)$ to be the maximum level of $B_i$ such that all buffers between $M_{i+1}$ and $M_j$ can become full at the same time. Alternatively, we can think of $l_j(i)$ as the maximum level of $B_i$ such that the failure of $M_j$ can cause $M_{i+1}$ to become blocked. It is

$$l_j(i) = N^p - \Psi(i + 1, j) \quad (3)$$

Note that $l_j(i)$ can assume values ranging from less than zero to greater than $N_i$ depending on the population and buffer sizes. Here, we focus on cases where $0 < l_j(i) < N_i$. To deal with these cases, [17] proposes a more detailed building block and a new set of decomposition positions. This approach is accurate and has been implemented for three-machine loops with certain restrictions on population and buffer sizes. However, Maggio’s building block can only take a single threshold into account. It would be possible to extend the method to larger loops, but the building block would have to become very complex to deal with multiple thresholds.

Maggio shows that blocking thresholds and starving thresholds are the same [17].
A.5 Loop Transformation

It is possible to eliminate the complications in the two-machine building blocks due to thresholds by using a transformation procedure. The transformation allows us to evaluate much larger loops for a wider range of population levels and buffer sizes than is possible using the method presented in [17]. Instead of dealing with the thresholds directly, we transform the loop into one without thresholds that behaves in almost the same way. The resulting loop is relatively easy to analyze.

Consider again the three-machine loop with buffers of size 5 and population 7. Into each of the three buffers, we insert a perfectly reliable machine so that the buffer of size 5 is replaced by an upstream buffer of size 3 and a downstream buffer of size 2 (see Figure 6). The performance of this new six-machine loop is approximately the same as the original three-machine loop, but we have eliminated all thresholds between zero and \( N_i \).

We can extend this approach to any \( K \)-machine loop. For each threshold \( 0 < l_k(i) < N_i \), we insert a perfectly reliable machine \( M_{k^*} \) into buffer \( B_i \) such that \( \Psi(k^*, k) = N^p \). \( B_i \) is now represented as a buffer of size \( N_i - l_k(i) \) followed by \( M_{k^*} \) followed by a buffer of size \( l_k(i) \). Since each unreliable machine can cause at most one threshold between zero and \( N_i \), the transformed loop will consist of at most 2\( K \) machines. Although the loop is larger, we can now use the same building block that is used in Tolio’s transfer line decomposition. Furthermore, the computational complexity does not increase with the addition of the new machines because no new failure modes are introduced.

A.6 Fixed Population Considerations

Once the loop is transformed to eliminate all thresholds between zero and \( N_i \), we must account for the limited propagation of blocking and starvation due to a fixed population level. To do this, we define the range of starvation and range of blocking, indexed on the buffer number. The range of starvation of \( B_i \) is the set \( \{ M_{s(i)}, M_{s(i)+1}, ..., M_i \} \), where \( M_{s(i)} \) is the machine farthest upstream which can cause \( B_i \) to become empty if it is

failed for a long period of time. Similarly, the range of blocking of \( B_i \) is the set \( \{ M_{i+1}, M_{i+2}, ..., M_{b(i)} \} \), where \( M_{b(i)} \) is the machine farthest downstream which can cause \( B_i \) to become full. These ranges are illustrated in Figure 7.

\[
M_{s(i)} = \min_j M_{i+j} \quad \text{s.t.} \quad \Psi(i, i+j) > N^p \quad (4) \\
M_{b(i)} = \max_j M_{i+j+1} \quad \text{s.t.} \quad \Psi(i+1, i+j+1) < N^p \quad (5)
\]

The loop population is incorporated into the model by including the building blocks only those virtual failure modes related to machines within the range of blocking and range of starvation. \( M^p(i) \) has virtual failure modes corresponding only to the failure modes of \( M_{s(i)} \) through \( M_{i-1} \). Likewise, \( M^d(i) \) has virtual failure modes corresponding to \( M_{i+2} \) through \( M_{b(i)} \).

**Simultaneous Blocking and Starvation**

If \( \Psi(v, w) = N^p \) then machine \( M_v \) can become simultaneously blocked and starved when \( M_w \) is down for a long period of time. This is the case where the thresholds \( l_v(v-1) = 0 \) and \( l_w(v) = N_v \). In transformed loops, this situation can occur at each reliable machine \( M_k \), when \( M_k \) fails since the buffer sum between the two machines \( \Psi(k^*, k) = N^p \) by construction.

The two-machine building block developed in [20] does not account for the states where both machines are down and the buffer level is either zero or full. Rather than modifying the building block, we associate the zero buffer level case with an upstream failure and the full buffer case with a downstream failure.

That is, when \( M_{k^*} \) is simultaneously blocked and starved by the failure of \( M_k \), we must consider the states of the two-machine lines associated with the buffers immediately upstream and downstream of \( M_{k^*} \). In the upstream two-machine line, the buffer is full. Both machines are down due to the failure of \( M_k \), but we treat the first machine as though it is up. This is because the second machine must have failed before

---

\(^2\)Note that the inequalities are strict. We use this convention to deal with the situation of simultaneous blocking and starvation.
the first one; otherwise the buffer could not be full. The failure of the first machine cannot affect the buffer level.

Similarly, in the two-machine line associated with the buffer downstream of $M_k$, we treat the second machine as though it is up when the failure of $M_k$ causes the buffer to be empty.

B. Loop Decomposition

In order to decompose closed-loop systems, we must first establish a building block and then find a way to relate these building blocks to one another. This section discusses the required parameters and the equations that we use to find them. Since it is always possible to transform a loop into one in which thresholds are not needed, we restrict our attention to loops without thresholds.

B.1 The Building Block Parameters

As in the Tolto transfer line decomposition, we evaluate the loop by breaking it up into a series of two-machine building blocks. Each building block $L(i)$ is associated with the buffer $B_i$ in the original loop. The upstream machine $M^u(i)$ has real failure modes corresponding to those of $M_i$, and virtual failure modes corresponding to those of machines $M_{i+1}$ through $M_{i-1}$. Similarly, $M^d(i)$ has real failure modes corresponding to those of $M_{i+1}$ and virtual failure modes corresponding to those of machines $M_{i+2}$ through $M_{i+1}$.

To evaluate the performance measure of the loop, we must find the virtual failure probabilities $p^u_{kj}(i)$ and $p^d_{kj}(i)$ for each $L(i)$. This is the objective solving the decomposition equations.

B.2 Decomposition Equations

The decomposition equations are nearly identical to the transfer line decomposition equations presented in [19]. In fact, we need only modify the indices to account for the range of blocking and starvation and the fact that loops contain as many buffers as machines.

III. NUMERICAL RESULTS

A. Performance of the Method

The method was tested extensively on three- to ten-machine loops with machine parameters and buffer sizes generated randomly using Microsoft Excel. Repair probabilities for each machine in the loop were drawn from a uniform distribution between 0.2 and 0.002 and were specified to be of the same order of magnitude. Failure probabilities were randomly generated from a uniform distribution such that the isolated efficiency $r/(r+p)$ of each machine in the loop was between 60 and 90 percent. Buffer sizes were drawn from a uniform distribution between $1/3r$ and $5/r$. For each loop, the decomposition and simulation were performed for all possible population levels. Here, we summarize the accuracy and convergence reliability. Details can be found in [13].

A.1 Accuracy

The method gives extremely accurate (error of less than 1 percent) approximations of average throughput when the number of parts (or holes) is greater than the number of machines and/or the size of the smallest buffer. Average buffer level errors were usually less than 1 percent in this range, but in some cases the errors were as high as 6 percent.

A.2 Convergence Reliability

In all cases studied, the decomposition algorithm converged. The criterion used for convergence was that the maximum difference in the value of all $E(i)$s between successive iterations be less than the specified tolerance of $10^{-6}$.

B. Observations on Loop Behavior

Consider a three-machine loop with buffers of size 10, 5, and 22 (see Figure 8). The production rate and average buffer levels are shown in Figures 9 and 10. Note the symmetry and flatness of Figure 9.

Fig. 8. Example of loop with transfer line flatness

IV. CONCLUSIONS AND FUTURE WORK

The purpose of this research was to build on Maggio’s work [17] to find a more practical general approach to evaluating closed-loop systems. Our transformation algorithm significantly reduces the complexity of large loops by eliminating multiple thresholds. The transformation and decomposition technique described in this paper provide extremely accurate approximations of average production rate.

There are several extensions to the method which would prove useful:
1. The approach described here could be extended to multiple loop systems. This is of particular interest for evaluating the performance of systems operated under token-based control policies. See [12].
2. The method could also be modified to deal with closed-loop systems in which multiple part types share
a common set of resources. In this type of system, different part types compete for resources and therefore the production of one part interferes with the production of another.

3. Another possibility is the combination of the first two items. The method can be extended to evaluate multiple loops with multiple part types.

4. The method should be extended to other models of production loops, including exponential processing time and continuous material models [6].

ACKNOWLEDGMENTS

We are grateful for research support from the Singapore-MIT Alliance, the Lean Aerospace Initiative, and the National Science Foundation. We are grateful for suggestions by Raniero Levantesi and Youichi Nonaka.

REFERENCES


