Modeling and Analysis of Manufacturing Systems with Multiple-Loop Structures

Zhenyu ZHANG[1] and **Stanley B. GERSHWIN**[1]
zhyzhang@mit.edu gershwin@mit.edu
[1] Massachusetts Institute of Technology

***Abstract*— Kanban and Constant Work-In-Process (CONWIP) control methods are designed to impose tight controls over inventory, while providing a satisfactory production rate. This paper generalizes systems with kanban or CONWIP control as assembly/disassembly networks with multiple-loop structures. We present a stochastic mathematical model which integrates the control information flows with material flows. Graph theory is used to analyze the multiple-loop structures. An efficient analytical algorithm is developed for evaluating the expected production rate and inventory levels. The performance of the algorithm is reported in terms of accuracy, reliability and speed.**

***Index Terms*—CONWIP, decomposition, graph theory, kanban, loop.**

I. Introduction

The study of kanban controlled systems can be traced back to the Toyota Production System in the 1950s. The classic kanban controlled system was designed to realize Just-In-Time (JIT) production, keeping a tight control over the levels of individual buffers, while providing a satisfactory production rate (Figure 1(a)).

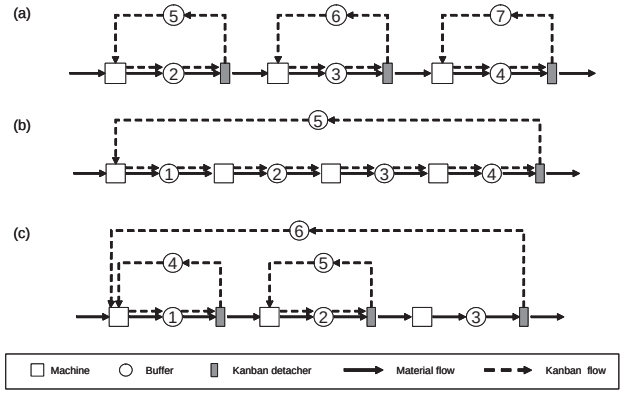

Fig. 1. Variations of kanban systems
(a) Classic kanban controlled system; (b) CONWIP controlled system; (c) Hybrid controlled system

There are several variations of kanban control widely used in industry, such as CONWIP control and hybrid control (Figure 1(b)(c)).

A. Essence of Kanban Control

Consider the kanban loop in Figure 1(a). Once a part enters a closed loop, a kanban card is attached to it. The kanban card is detached from the part when it leaves the closed loop and proceeds to the next stage. The number of kanbans within the closed loop is constant. We define it as the *invariant* of the loop. Similarly, when we look at the CONWIP loop in Figure 1(b), kanban cards are attached to the parts at the first stage of the production line while they are detached from the parts at the last stage. The total number of kanbans circulated within the CONWIP loop gives the loop invariant β:

$$\beta = b(1,t) + b(2,t) + b(3,t) + b(4,t) + b(5,t) \qquad (1)$$

in which $b(i,t)$ is the level of B_i at time t.

The invariant imposes an upper limit of the buffer levels within the closed loop. For example, the total number of parts N allowed in the large CONWIP loop is constrained by:

$$N = b(1,t) + b(2,t) + b(3,t) + b(4,t) \leq \beta \qquad (2)$$

More generally, systems using kanban controls can be represented as a set of systems with multiple-loop structures. Each closed loop has a loop invariant.

B. Importance of Multiple-Loop Structures

To control a given production system, a variety of kanban control methods can be used. Classic kanban control, CONWIP control and hybrid control are compared by Bonvik [1]. The hybrid control method is demonstrated to have the best inventory control performance. Therefore, to study the design of control structures is valuable for developing insights into operational control.

After we determine the control method, the design parameters of the closed loop, such as the number of kanbans, also have significant effects on the system's performance and cost. When CONWIP control is implemented by circulating pallets instead of kanbans, as pallets cost money and take up space, the optimal selection of control parameters, such as the number of pallets and the storage buffer space of the pallets, could provide a saving of millions of dollars.

Therefore, an exhaustive study of the properties of multiple-loop structures is needed. This study is challenging but highly valuable. It will provide a theoretical basis and practical guidelines for factory design and operation.

C. Relevant Work

As the analysis and design of kanban systems usually involve evaluating a large number of variations with different structures and parameters, analytical methods are much more promising in terms of computational efficiency. Markov chains and decomposition method are used to evaluate stochastic manufacturing systems by breaking them down into a set of two-machine lines (building blocks) [2], [3]. These building blocks can be evaluated analytically by using the method in [8]. This method models the two-machine lines by assigning multiple failure modes to the pseudo-machines, instead of using single failure mode. This new decomposition method was applied to study systems with the closed loop. Werner and Gershwin [5], [10] developed an efficient method to evaluate large single-loop systems. Levantesi [6] extended it to small multiple-loop systems. However, Levantesi's method demonstrated the feasibility of his approach but required an analysis of the propagation of blocking and starvation that had a very inefficient method for doing this. Therefore, this method is not able to provide satisfying speed and reliability while evaluating large-scale multiple-loop systems. In addition, a systematic understanding of the behavior of multiple-loop structures has not been developed yet.

D. Research Goal and Contributions

This research is intended to investigate the behavior of multiple-loop structures, to develop mathematical models and efficient analytical methods to evaluate system performances, and to help design factories. Specifically, the contributions of this paper include:

- a unified model to represent multiple-loop structures which integrates information flows with material flows
- a systematic method to analyze blocking and starvation propagation based on graph theory
- an efficient algorithm to evaluate assembly/disassembly systems with arbitrary topologies
- experiments to demonstrate that the algorithm is accurate, reliable, and fast.

E. Outline

In Section II we introduce assembly/disassembly networks with multiple-loop structures. To analyze blocking and starvation propagation in systems with complex topologies, in Section III, we develop a graph model. In Section IV, blocking and starvation analysis is discussed based on the graph model. In Section V, we present an efficient algorithm for evaluating assembly/disassmebly networks with arbitrary topologies. We also discuss the performance of the algorithm in terms of accuracy, reliability and speed. Conclusions and future research are in Section VI.

II. Assembly/Disassembly Networks with Multiple-Loop Structures

In this section, we develop a model to provide an integrated view of material flows and information flows.

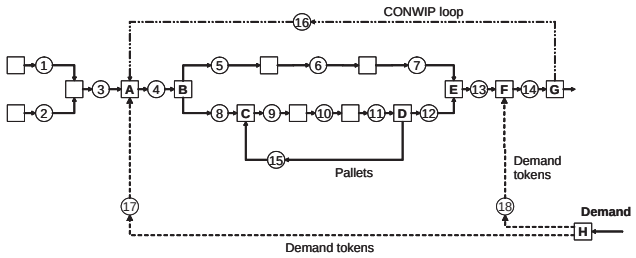

Fig. 2. Assembly/disassembly network with multiple-loop structures

A. A Unified Model

Consider the system in Figure 2. The machines in the system are only allowed to perform single part, assembly or disassembly operations (We do not include merges or spits of flows). A CONWIP control loop is implemented between machine G and A. Demand tokens are generated by machine H to make base stock control at machine A and F.

Various meanings of assembly/disassembly operations are represented in the unified graph of Figure 2. Traditionally, assembly/disassembly is used to describe a process involving two or more real work pieces. However, this is not always the case. Sometimes, a work piece is assembled to a pallet or fixture when it enters a system. After going though a set of processes, the work piece is disassembled from the pallet. Therefore, assembly/disassembly takes place between a real work piece and a fixture or pallet. For example, machine B and E in Figure 2 conduct the disassembly and assembly of real parts. Machine C and D disassemble and assemble real parts with pallets.

When we consider kanban control, the attach and detach operations between kanban and work pieces are actually assembly/disassembly operations. By imagining the kanban as a medium which conveys control information, the assembly/disassembly operations happen between information flows and material flows.

More interestingly, the demand information can be embedded into a tandem line with material flow [4] by using virtual assembly/disassembly machines. In Figure 2, machine H is a demand machine which generates demand token flows by disassembly. The demand tokens are assembled with real parts at machine A and F.

In summary, the definition of assembly/disassembly is extended to include both material flows and information flows. Kanban, CONWIP and other information-based control methods can be modeled as assembly/disassembly networks with multiple-loop structures.

B. Approach to Evaluation

Decomposition is developed as an approximation technique to evaluate large-scale complex manufacturing systems [2]. A two-machine one-buffer line is designed as the building block to approximate the behavior of the flow in the buffer. The upstream pseudo-machine of the two-machine line approximates

the failure propagation from the upstream, while the downstream pseudo-machine approximates the failure propagation from the downstream.

By using the multiple failure mode method to decompose the system [9], the assignment of the failure modes is determined according to the blocking and starvation propagation property. If any machine failure could cause the observed buffer to be full, its immediate upstream machine is blocked and thus the failure is assigned to the downstream pseudo-machine. Likewise, any machine failure which could result in an empty buffer is assigned to the upstream pseudo-machine.

To obtain the blocking and starvation propagation property, we are interested to know the interrelationship between machine failures and buffer levels: given a machine failure, what are the levels of buffers in the system and which machines in the system could be blocked or starved. Therefore, before we decompose the system into building blocks, we should derive the property of blocking and starvation propagation.

While there are closed loops in the system, the behavior of blocking and starvation propagation become complicated. The analysis of blocking and starvation of multiple-loop structures is the focus of this paper. Decomposition evaluation will be briefly mentioned since it is almost the same as existing techniques [5], [10].

C. Loop Invariants

Loop invariants result in the complicated behavior of blocking and starvation propagation. At this point, we introduce the definition of loop invariants.

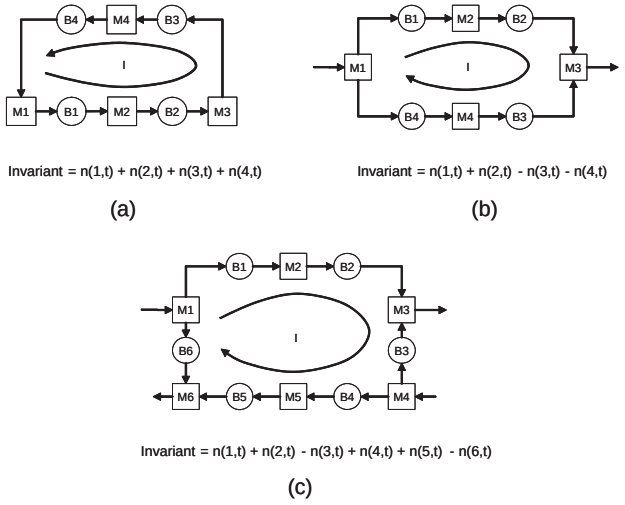

Fig. 3. Loop invariant of single-loop systems

Consider the variation of the single loop in Figure 3(b). Although there is no part circulated in this disassembly/assembly system, an invariant still can be found. Consider the disassembly at machine 1. Whenever one part goes into the upper branch, there must be one part going into the lower branch. Similar things happen to the assembly at machine 2. Therefore, the difference between the sum of the buffer levels of the upper branch and that of the lower branch is constant.

A more complicated variation of single loop is shown in Figure 3(c). The flow direction changes four times in the loop. Loop direction, as an abstract concept, is introduced for the purpose of defining the invariant. An invariant can be then obtained by adding the levels of buffers whose direction agree with the loop direction and subtracting the levels of all the rest.

III. Graph Model of Assembly/Disassembly Networks

This section presents, in the manufacturing system context, how to apply graph theory to develop a graph model for the analysis and design of large-scale complex assembly/disassembly networks with multiple-loop structures.

A. Terminologies and Notations

As the terminologies of graph theory vary from one book to another, in order to keep consistency, the definition of terminologies used in this paper are referred from [7].

The vertices and arcs of a digraph are denoted by $v_i, i = 1, 2, ..., n$ and $a_j, j = 1, 2, ..., m$. The topology of the digraph could be represented by the all-vertex incidence matrix $\Phi = [\phi_{ij}]$. A set of fundamental circuits in the digraph is specified by $c_k, k = 1, 2, ..., m-n+1$. The circuit matrix corresponding to the set of fundamental circuits is given by $\Psi = [\psi_{kj}]$.

B. Graph Model

1) Underlying Digraph: For any pair of machines in an assembly/disassembly network, a buffer is used to model the interstage connection if there exists a flow between the pair of machines. By convention, a machine is allowed to handle multiple flows of parts by assembly/disassmebly, whereas a buffer is only allowed to process a single incoming flow from its upstream machine and generate a single outgoing flow toward its downstream machine. In such a way, a buffer could be defined exactly as an arc in graph theory,

$$B_j = (u(j), d(j)) \tag{3}$$

where $u(j)$ and $d(j)$ denote the upstream and downstream machines of B_j, respectively.

Machines in the network are mapped to vertices in the digraph. For a machine M_i, we define $U(M_i)$ as the set of its upstream buffers, and $D(M_i)$ as the set of its downstream buffers. The initial vertex of an arc is the upstream machine of the corresponding buffer, while the terminal one is the downstream machine. Loops in assembly/disassembly networks are essentially defined in the same way as circuits in digraphs. Each loop corresponds to a circuit vector of the digraph.

In summary, any assembly/disassembly network N has a unique underlying connected digraph G. The vertex set V and arc set A correspond to the set of machines M and the set of buffers B, respectively. The set of circuits C is identical to the set of loops L. Therefore, network $(M, B, L)_N$ is equivalent to $(V, A, C)_G$. For example, the underlying digraph of the network in Figure 2 is given in Figure 4

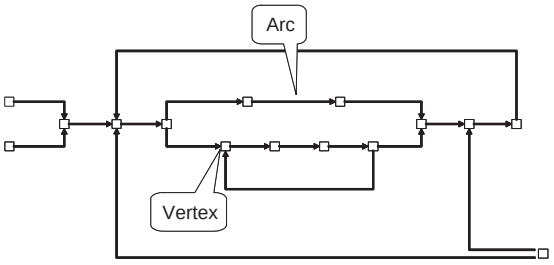

Fig. 4. Underlying digraph of an assembly/disassembly network

2) Capacity, Level and Flow: In an assembly/disassmebly network N, the *capacity* of an arc, or buffer, is defined as the size of buffer N_j. The *level* of B_j is an assignment of a non-negative function $b(j,t)$, which denotes the amount of material stored in B_j at time point t subject to the *capacity constraint*

$$0 \leq b(j,t) \leq N_j \qquad \text{for all } B_j \in B \tag{4}$$

The *flow* or the amount of materials transported through M_i during the time period $(0,t)$ is denoted by $q(i,t)$.

3) Machine Failure, Blocking and Starvation: When a machine M_i is down due to a failure at time $t=0$, the flow transported through the machine is forced to be zero:

$$q(i,t) = 0 \qquad t \geq 0 \tag{5}$$

As a result, the upstream buffers of M_i tend to be filled up, while the downstream buffers of M_i will be depleted gradually.

For a given machine M_i in the system, suppose any of its downstream buffers is full,

$$b(v,t) = N_v \qquad v \in U(M_i) \tag{6}$$

M_i is said to be *blocked* at time t through buffer B_v.
Similarly, M_i is *starved* at time t if any of its upstream arcs is empty:

$$b(w,t) = 0 \qquad w \in D(M_i) \tag{7}$$

C. Conditions

1) Conservation Condition: Suppose at time $t=0$, the initial inventory level of B_j is $b(j,0)$. Recall that the amount of material transported through M_i during the time period $(0,t)$ is denoted by $q(i,t)$. The resultant inventory level of B_j at time t is given by

$$b(j,t) = b(j,0) + q(u(j),t) - q(d(j),t) \tag{8}$$

in which $u(j)$ and $d(j)$ are, respectively, the upstream and downstream machines of B_j.

This equation is called the *conservation condition* which requires that, during a time period, the increase or decrease of the buffer level is equal to the net flow of materials entering or leaving the buffer.

By using the corresponding column vector $\mathbf{\Phi_{*j}} = [\phi_{1j}, \phi_{2j}, \ldots, \phi_{nj}]^T$ of incidence matrix Φ, we can rewrite the conservation condition for each buffer as follows,

$$b(j,t) = b(j,0) + [\phi_{1j}, \ldots, \phi_{nj}]\,[q(1,t), \ldots, q(n,t)]^T \qquad \text{for all } B_j \in B \tag{9}$$

Define the *level vector* of buffer set B

$$\mathbf{b}(t) = [b(1,t), b(2,t), \ldots, b(m,t)]^T$$

Define the *flow vector* during $(0,t)$ of machine set M

$$\mathbf{q}(t) = [q(1,t), q(2,t), \ldots, q(n,t)]$$

The equations of the conservation condition can be concisely written as

$$\mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) \qquad t \geq 0 \tag{10}$$

2) Invariance Condition: When there exists loops or circuits in assembly/disassembly network N with n machines and m buffers, we can obtain another set of equations called the *invariance condition*. Fundamental circuit matrix Ψ_f is used to depict the condition.

The *circuit value* of each fundamental circuit vector L_k is defined by

$$\beta_k = [\psi_{k1}, \ldots, \psi_{km}]\,[b(1,t), \ldots, b(m,t)]^T \qquad k = 1, 2, ..., n-m+1 \tag{11}$$

As the circuit value is an invariant of time, the above equation is the invariance condition of loop L_k. The circuit values of a set of fundamental circuits or loops can be organized into a column vector β:

$$\beta = [\beta_1, \beta_2, \ldots, \beta_{m-n+1}]$$

The invariance condition with respect to the set of fundamental circuits can be written as:

$$\Psi_f \mathbf{b}(t) = \beta \qquad t \geq 0 \tag{12}$$

IV. Blocking and Starvation Analysis

In this section, we apply the graph model to analyze the blocking and starvation propagation in complex assembly/disassmebly networks.

A. Assumption

With the purpose to understand the maximal effect of a single machine failure on the propagation of blocking and starvation, we must make the *single machine failure* assumption: In an assembly/disassembly network, once there is a failure at machine M_i, none of the other machines is allowed to fail. The failure of M_i should persist for a sufficient amount of time such that the levels of all the buffers remain in the steady values.

This assumption, though not necessary to be true in real cases, is very helpful to investigate the maximal effect of propagation due to a single machine failure and eliminate the effects caused by other failed machines. Under this assumption, suppose M_i failed at time $t = 0$, the buffer level vector in the steady state with respect to the single machine failure at M_i is defined as below:

$$\mathbf{b}(+\infty|M_i) = \lim_{\substack{q(i,t)=0 \\ t\to+\infty}} \mathbf{b}(t) \tag{13}$$

in which $q(i,t)$ is defined as the amount of material transported through machine M_i during time period $(0,t)$.

In the assumption, *steady state* is referred to that, when there is a single machine failure at M_i, the propagation of blocking and starvation has reached the maximal effect after a sufficient amount of time such that

- The buffer levels will not change with time and remain in the steady values.
- There is no flow through any machine in the system.
- Except the failed machine, other machines can only be in one of the three states: being blocked, being starved or being simultaneously blocked and starved. Blocking and starvation of a machine are defined as follows:

$$M_k \in M, k \neq i \quad \text{is} \begin{cases} \text{blocked} & \text{if} \quad \exists j \in D(M_k) \\ & \quad b(j,+\infty) = N_j \\ \text{starved} & \text{if} \quad \exists j \in U(M_k) \\ & \quad b(j,+\infty) = 0 \end{cases} \tag{14}$$

B. Objective

The objective of blocking and starvation analysis is formulated as solving the following problem in the graph model context:

$$\begin{array}{ll} \text{Given} & q(i,t) = 0 \qquad t \geq 0, M_i \in M \\ \text{Solve} & \mathbf{b}(+\infty|M_i) \end{array} \tag{15}$$

in which M_i is the machine failed at time $t = 0$.

Intuitively, $\mathbf{b}(+\infty|M_i)$ is the solution $\mathbf{b}(t)$ of the multi-objective optimization problem as follows:

$$\begin{array}{ll} \text{maximize} & \mathbf{q}(t) = [q(1,t), q(2,t), \cdots, q(n,t)]^T \quad (16) \\ \text{subject to} & \mathbf{b}(t) = \mathbf{b}(0) + \Phi^T \mathbf{q}(t) \quad (17) \\ & \Psi_f \mathbf{b}(t) = \beta \quad (18) \\ & 0 \leq \mathbf{b}(t) \leq \mathbf{N} \quad (19) \\ & q(i,t) = 0 \quad (20) \end{array}$$

where $\mathbf{N}$ is defined as the *buffer size vector*:

$$\mathbf{N} = [N_1, N_2, \ldots, N_m]$$

The objective is to maximize n-dimensional vector $\mathbf{q}(t)$. Constraints (17) and (18) are the conservation and invariant conditions of the graph model. This optimization problem gives an equivalent statement because, in the steady state with respect to any single machine failure, the amount of material through each machine must be maximized when $t \to +\infty$ and thus there is no flow through any machine in the system. The solution of the optimization problem $\mathbf{b}(t)$ is unique and regardless to the initial condition $\mathbf{b}(0)$. This indicates that the blocking and starvation propagation property is deterministic.

C. 'Machine Failure - Buffer Level' Matrix

A matrix Θ is designed to neatly represent the blocking and starvation propagation property between machine failures and buffer levels under single machine failure assumption. Consider the five-machine tandem line in Figure 5, all the buffers has size 10. The 'machine failure - buffer level' matrix of this tandem line is given as below:

$$\Theta = \begin{array}{c|cccc} & B_1 & B_2 & B_3 & B_4 \\ M_1 & 0 & 0 & 0 & 0 \\ M_2 & 10 & 0 & 0 & 0 \\ M_3 & 10 & 10 & 0 & 0 \\ M_4 & 10 & 10 & 10 & 0 \\ M_5 & 10 & 10 & 10 & 10 \end{array}$$

Fig. 5. A five-machine tandem line

In the matrix, the vertical dimension represents the failures of different machines, while the horizontal one records the levels of different buffers in steady state. Entry θ_{ij} reflects the level of B_j in the steady state with respect to the single machine failure at M_i:

$$\theta_{ij} = b(j,+\infty|M_i) \qquad M_i \in M, B_j \in B \tag{21}$$

In the matrix, column j represents the levels of B_j with respect to the failures of different machines. By looking into row i in matrix Θ, we can identify buffer level vector $\mathbf{\Theta_{i*}}$ with respect to the single machine failure at M_i. This matrix tool efficiently records the results of blocking and starvation analysis. It could be used to derive the information for decomposition evaluation.

D. Induction Method

The most critical part in blocking and starvation analysis is to obtain the buffer level vectors in steady state. Theoretically, we could solve the buffer level vectors in steady state from optimization problem (16-20). However, directly solving this multi-objective optimization problem is very difficult and inefficient, especially when the system is large-scale and multiple loops are coupled together to form a complex topology. Therefore, we are seeking an efficient and intuitive approach to solve the buffer level vectors in steady state.

In this section, we present the induction method to efficiently solve the buffer level vector in steady state for assembly/disassmebly networks with arbitrary topologies.

1) Intuition: For any assembly/disassmebly network N with n machines and m buffers, we can establish a unique underlying connected digraph G with n vertices and m arcs. By choosing a spanning tree T of G, we can get a set of $m-n+1$ chords of T. The digraph G can be constructed by adding chords $c_1, c_2, ..., c_{m-n+1}$ to spanning tree T:

$$G = T \cup c_1 \cup c_2 \cup ... \cup c_{m-n+1} \tag{22}$$

Therefore, the system can be constructed by selecting a tree-structured network corresponding to a spanning tree T of G and adding a set of buffers corresponding to chords $c_1, c_2, ..., c_{m-n+1}$ of T.

Consider the four-loop assembly/disassembly network G shown in Figure 2. The digraph T of Figure 6(a) is a spanning tree of G. A set of subsystems of G with respect to spanning tree T is shown in Figure 6(b)(c)(d).

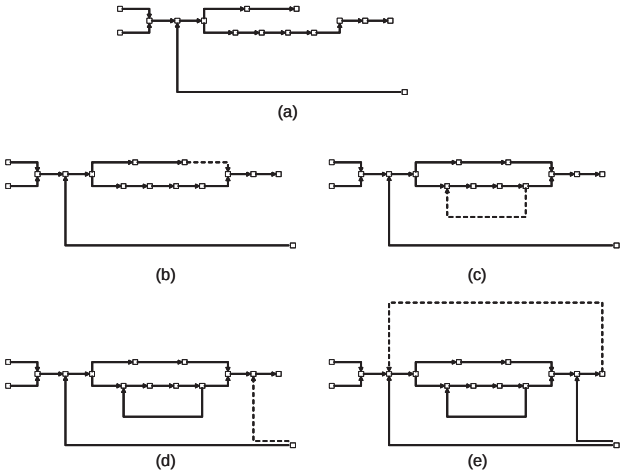

Fig. 6. A spanning tree and corresponding subsystems and chords (indicated by dashed lines)

Notice that directly solving the steady state buffer level vectors of a tree-structured network is easy. Therefore, we develop the intuition of induction method as follows:

- Construct the 'machine failure - buffer level' matrix for the tree-structured network. Given a B_j in the tree, if the failed machine M_i is connected to B_j via the downstream machine of B_j, then $\theta_{ij} = b(j, +\infty | M_i) = N_j$; If the failed machine M_i is connected to B_j via the upstream machine of B_j, then $\theta_{ij} = b(j, +\infty | M_i) = 0$.
- Add a buffer to form a new loop. Thus the matrix will be expanded by one column, which corresponds to the added buffer. The induction step is to solve the steady state buffer level vectors in the matrix based on those in the previous matrix.

2) Induction Sequence: For an assmebly/disassembly network $N = (M, B, L)$, the selection of tree-structured network T is not arbitrary because the set of fundamental loops L is usually specified with its components and invariant. Therefore, in the underlying digraph G, we need to determine the set of chords and spanning tree corresponding to the set of fundamental loops.

Recall that the components of the set of fundamental loops L can be represented by the circuit matrix Ψ. Each row vector $\mathbf{\Psi_{k*}}$ of Ψ corresponds to a fundamental loop L_k. Each fundamental loop has a defining chord. Therefore, in row k, we can identify at least one column c_k which gives

$$|\psi_{k,c_k}| = \sum_{i=1}^{m-n+1} |\psi_{i,c_k}| = 1 \tag{23}$$

Notice that B_{c_k} corresponds to chord c_k of loop L_k. Therefore, among the set of buffers B, a subset $B_c = \{B_{c_1}, B_{c_2}, ..., B_{c_{m-n+1}}\}$ corresponding to the set of fundamental loops $L_k, k = 1, ..., m-n+1$ could be identified. The set of machines M and the remaining $n-1$ buffers compose a spanning tree T which is equivalent to the tree-structured network.

$$T \equiv (M, B - B_c)_N \tag{24}$$

Let the tree-structured network as Ω_0 and define a set of subsystems $\Omega_k, k = 1, 2, ..., m-n+1$ of network N:

$$\Omega_k = \begin{cases} (M, B - B_c)_N & k = 0 \\ \Omega_{k-1} \cup B_{c_k} & k = 1, ..., m-n+1 \end{cases} \tag{25}$$

The subsystems $\Omega_0, \Omega_1, ..., \Omega_{m-n+1}$ specify the induction sequence to construct the system starting from a tree-structured network. In kth induction step, B_{c_k} will be added and the invariant condition with respect to loop L_k should be satisfied.

Finally, we reorder the set of buffers such that

$$\begin{aligned} &B_1, B_2, ..., B_{n-1} \in \Omega_0 \\ &B_{n-1+k} = B_{c_k} \qquad k = 1, 2, ..., m-n+1 \end{aligned} \tag{26}$$

3) Induction Operator: During induction, suppose we have constructed the 'machine failure - buffer level' of subsystem Ω_k. Denote the matrix by $\Theta(\Omega_{k-1})$, then entry $\theta_{ij}(\Omega_{k-1})$ is defined as

$$\begin{aligned} &\theta_{ij}(\Omega_{k-1}) = b(j, +\infty | M_i)_{\Omega_{k-1}} \\ &i = 1, ..., n; j = 1, ..., n-2+k \end{aligned} \tag{27}$$

The buffer level vector in the steady state with respect to the single machine failure at M_i is a row vector of $\Theta(\Omega_{k-1})$ given as below:

$$\begin{aligned} \mathbf{\Theta_{i*}}(\Omega_{k-1}) = [&\theta_{i1}(\Omega_{k-1}) \quad \theta_{i2}(\Omega_{k-1}) \\ &\cdots \quad \theta_{i(n-2+k)}(\Omega_{k-1})] \end{aligned} \tag{28}$$

When we add a new B_{n-1+k} to subsystem Ω_{k-1}, the new system Ω_k will have one more loop L_k whose chord is c_k. The 'machine failure - buffer level' matrix $\Theta(\Omega_k)$ will have one more column which corresponds to B_{n-1+k}, originally B_{c_k}. Therefore, each buffer level vector has one more component:

$$\mathbf{\Theta_{i*}}(\Omega_k) = [\theta_{i1}(\Omega_k) \quad \theta_{i2}(\Omega_k) \quad \cdots \quad \theta_{i(n-2+k)}(\Omega_k) \quad | \quad \theta_{i(n-1+k)}(\Omega_k)] \tag{29}$$

From the perspective of induction, the buffer level vectors of current subsystem Ω_k should be derived based on those of previous subsystem Ω_{k-1}. In addition, the invariant β_k should also be the input of induction operation since it determines whether all the components of $\mathbf{\Theta_{i*}}(\Omega_{k-1})$ are conservative or not. Let Γ be the induction operator, then we have

$$\mathbf{\Theta_{i*}}(\Omega_{k-1}) = \Gamma(\mathbf{\Theta_{i*}}(\Omega_{k-1}), \beta_k) \qquad i = 1, 2, ..., n \tag{30}$$

in which β_k is the invariant of loop L_k, whose chord is B_{n-1+k}.

Although all-vertex incidence matrix $\Phi(\Omega_{k-1})$ and circuit matrix $\Psi_f(\Omega_{k-1})$ are not considered as inputs, they are part of induction operation such that the conservation and invariant conditions are satisfied.

4) Solution Technique:

a) Insight: The focus of induction method is to design the induction operator Γ to solve the buffer level vectors correctly. Recall the optimization formulation in Section IV.B and compare the formulation of subsystem Ω_k and that of subsystem Ω_{k-1}. In kth step of induction, B_{n-1+k} is added to form L_k. Therefore, in the formulation of Ω_k, we have three additional equations as follows:

$$b(n-1+k,t) = b(n-1+k,0) + q(u(n-1+k),t) - q(d(n-1+k),t) \tag{31}$$

$$\left[\psi_{k1}, \psi_{k2}, \ldots, \psi_{k(n-1+k)}\right] \begin{bmatrix} b(1,t) \\ b(2,t) \\ \vdots \\ b(n-1+k,t) \end{bmatrix} = \beta_k \tag{32}$$

$$0 \le b(n-1+k,t) \le N_{n-1+k} \tag{33}$$

in which $u(n-1+k)$ is the upstream machine of B_{n-1+k} while $d(n-1+k)$ is the downstream machine of B_{n-1+k}. N_{n-1+k} is the size of B_{n-1+k}.

In the steady state with respect to the single machine failure at M_i in Ω_k, we conserve the value of components $\theta_{i1}(\Omega_{k-1}), ..., \theta_{i(n-2+k)}(\Omega_{k-1})$

$$b(j,t) = \theta_{ij}(\Omega_{k-1}) \qquad j = 1, ..., n-2+k \tag{34}$$

and relax the capacity constraint (33), then we can rewrite (32) and solve the buffer level of B_{n-1+k} as follows:

$$\left[\psi_{k1}, \psi_{k2}, \ldots, \psi_{k(n-1+k)}\right] \begin{bmatrix} \theta_{i1}(\Omega_{k-1}) \\ \theta_{i2}(\Omega_{k-1}) \\ \vdots \\ \theta_{i(n-2+k)}(\Omega_{k-1}) \\ b(n-1+k,t) \end{bmatrix} = \beta_k \tag{35}$$

$$b(n-1+k,t) = \left(\beta_k - \textstyle\sum_{s=1}^{n-2+k} \psi_{ks} \times \theta_{is}(\Omega_{k-1})\right) \Big/ \psi_{k(n-1+k)} \tag{36}$$

If $b(n-1+k,t)$ of (36) satisfies (33), the solution of system Ω_k is

$$\theta_{ij}(\Omega_k) = \begin{cases} \theta_{ij}(\Omega_{k-1}) & j = 1, ..., n-2+k \\ \left(\beta_k - \sum_{s=1}^{n-2+k} \psi_{ks} \times \theta_{is}(\Omega_{k-1})\right) \Big/ \psi_{k(n-1+k)} & j = n-1+k \end{cases} \tag{37}$$

If $b(n-1+k,t)$ of (36) does not satisfy (33), the level of B_{n-1+k} is either negative or larger than the size of B_{n-1+k}.

- When $b(n-1+k,t) < 0$, it seems that there is a backlog in B_{n-1+k} and the downstream machine of B_{n-1+k} overdrafts $-b(n-1+k,t)$ parts from B_{n-1+k}.
- When $b(n-1+k,t) > N_{n-1+k}$, it seems that there is an excess in B_{n-1+k} and the upstream machine of B_{n-1+k} overflows $b(n-1+k,t) - N_{n-1+k}$ parts to B_{n-1+k}.

b) Reverse Operation Policy: A reverse operation policy is developed to eliminate overdrafts or overflows. We firstly define some new quantities.

- *Nominal Level*: In the kth induction step, the nominal level of B_{n-1+k} is calculated according to (32) disregarding the capacity constraint.
- *Overflow and Overdraft*: Given a B_j in subsystem Ω_k,
  - When $0 \le b(j,t) \le N_j$, B_j has neither overflow nor overdraft;
  - When $b(j,t) > N_j$, B_j has overflow which is $b(j,t) - N_j$;
  - When $b(j,t) < 0$, B_j has overdraft which is $-b(j,t)$.

  Therefore, overflow and overdraft of B_j are defined as follows:

$$\delta_j^+ = \max\{b(j,t) - N_j, 0\} \tag{38}$$

$$\delta_j^- = \max\{-b(j,t), 0\} \tag{39}$$

Assume in the kth induction step, the single machine failure happens at M_i in subsystem Ω_k. We reuse the solution of the buffer levels in the $(k-1)$th induction

$$b(j,t) = \theta_{ij}(\Omega_{k-1}) \qquad j = 1, ..., n-2+k \tag{40}$$

and calculate the nominal level of B_{n-1+k} using (32). If there exists overflow δ^+_{n-1+k} in B_{n-1+k}, identify the upstream machine $M_{u(n-1+k)}$ of B_{n-1+k} and operate it reversely to process δ^+_{n-1+k} parts such that the levels of upstream and downstream buffers will be updated as follows:

- For the upstream buffers of $M_{u(n-1+k)}$:

$$b(v,t) = b(v,t) + \delta^+_{n-1+k} \qquad v \in U(M_{u(n-1+k)}) \quad (41)$$

- For the downstream buffers of $M_{u(n-1+k)}$:

$$b(w,t) = b(w,t) - \delta^+_{n-1+k} \qquad w \in D(M_{u(n-1+k)}) \quad (42)$$

Therefore, the updated level of B_{n-1+k} becomes N_{n-1+k}. B_{n-1+k} is full so that its upstream machine $M_{u(n-1+k)}$ is in steady state.

On the other hand, if there exists overdraft δ^-_{n-1+k} in B_{n-1+k}, identify the downstream machine $M_{d(n-1+k)}$ of B_{n-1+k} and operate it reversely to process δ^-_{n-1+k} parts such that the levels of upstream and downstream buffers will be updated as follows:

- For the upstream buffers of $M_{d(n-1+k)}$:

$$b(v,t) = b(v,t) + \delta^-_{n-1+k} \qquad v \in U(M_{d(n-1+k)}) \quad (43)$$

- For the downstream buffers of $M_{d(n-1+k)}$:

$$b(w,t) = b(w,t) - \delta^-_{n-1+k} \qquad w \in D(M_{d(n-1+k)}) \quad (44)$$

After the update, the overdraft in B_{n-1+k} is balanced. B_{n-1+k} becomes empty so that its downstream machine $M_{u(n-1+k)}$ is therefore in steady state.

In case that, for a given machine M_q, there exist multiple overflows among its upstream buffers or multiple overdrafts among its downstream buffers, we update the buffer levels as given below:

Let

$$\delta_q = \max\{\delta^+_v, \delta^-_w\} \qquad \text{for all } v \in U(M_q), w \in D(M_q) \quad (45)$$

- For the upstream buffers of M_q:

$$b(v,t) = b(v,t) + \delta_q \qquad v \in U(M_q) \quad (46)$$

- For the downstream buffers of M_q:

$$b(w,t) = b(w,t) - \delta_q \qquad w \in D(M_q) \quad (47)$$

c) Induction Operator: Finally, we summarize induction operator Γ. Assume in the kth induction step, the single machine failure happens at M_i in subsystem Ω_k:

- Assign buffer levels with the buffer level vector $\mathbf{\Theta_{i*}}(\Omega_{k-1})$ in subsystem Ω_{k-1} using (40).
- Calculate the nominal level, overflow and overdraft of buffers $B_j, j = 1, ..., n-1+k$ using (32), (38) and (39).
- Identify a set of machines M^* which has either overflow among its downstream buffers or overdraft among its upstream buffers.
- Calculate the reverse quantity δ_q of $M_q \in M^*$ using (45).
- Update the upstream and downstream buffers of M_q using (46) and (47).
- Check the steady state condition. If the system is not in steady state, repeat steps (1) to (4). If the system is in steady state, assign buffer level vector $\mathbf{\Theta_{i*}}(\Omega_k)$:

$$\theta_{ij}(\Omega_k) = b(j,t) \qquad j = 1, ..., n-1+k \quad (48)$$

Consequently, we are able to efficiently derive the 'Machine failure - Buffer level' matrix for complex assembly/disassembly networks with arbitrary topologies.

V. Algorithm and Results

A. Algorithm

We present a comprehensive algorithm to synthesize the blocking and starvation analysis and the decomposition evaluation to predict the performance of an assembly/disassembly network with arbitrary topology.

1) Phase I: Blocking and Starvation Analysis:

- Establish the graph model for assembly/disassembly system $N = (M, B, L)$ composed of n machines, m buffers and $m - n + 1$ loops.
- Identify $m - n + 1$ chords of the graph and create a set of subsystems $\Omega_k, k = 0, ..., m - n + 1$, in which Ω_0 is a spanning tree of N.
- For subsystem Ω_0, construct the 'machine failure - buffer level' matrix $\Theta(\Omega_0)$.
- Construct matrices $\Theta(\Omega_k), k = 1, ..., m - n + 1$ using induction method. In the kth induction step, assume the single machine failure at M_i, solve buffer level vector $\mathbf{\Theta_{i*}}(\Omega_k)$ using induction operator $\Gamma(\mathbf{\Theta_{i*}}(\Omega_{k-1}), \beta_k)$.

2) Phase II: Decomposition Evaluation: The procedures of decomposition evaluation include:

- Eliminate the buffer thresholds;
- Decompose the transformed system and setup the building blocks;
- Evaluate the unknown parameters and record the performance measures.

Details of each procedure could be referred to [5], [10].

B. Experiment Design

We experiment the algorithm by evaluating systems with 15 machines and 4 loops. As we are interested to know how well the algorithm performs while evaluating systems with different complex structures, we should randomly generate both the system structure and the parameters of the machines and buffers. Therefore, we generate 50 random system structures. For each system structure, we generate 50 sets of parameters. Therefore, we have 1500 cases in total.

To validate the accuracy of the algorithm, we compare the results with simulation. For production rates, we calculate the percent error of the approximated production rate from the simulated production rate as follows:

$$Error_{PR} = \frac{PR_{decomp} - PR_{sim}}{PR_{sim}} \times 100\% \quad (49)$$

For average buffer levels, we calculate the average absolute percent error of the approximated average buffer level as follows:

$$Error_{BL} = \frac{\sum_{j=1}^{m} \frac{|\bar{b}(j)_{decomp} - \bar{b}(j)_{sim}|}{N_j/2}}{m} \times 100\% \qquad (50)$$

C. Numerical Results

We discuss the algorithm performances in terms of accuracy, reliability, and speed.

1) Accuracy: The percent errors calculated for 1500 cases are shown in Figure 7 and 8. The mean of the absolute percent errors of approximated production rate is 0.88%. The mean of the average absolute percent errors of approximated buffer levels is 7.93%.

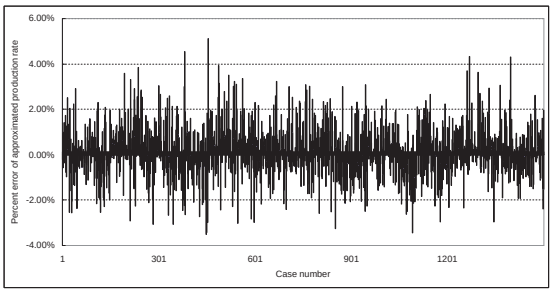

Fig. 7. The errors in the decomposition approximation for production rate

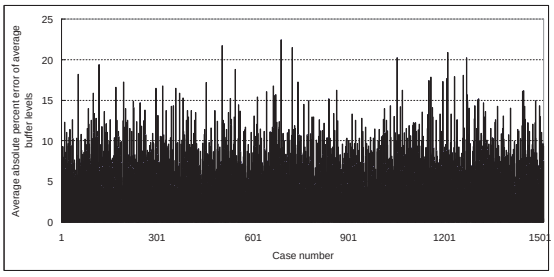

Fig. 8. The errors in the decomposition approximation for buffer levels

2) Reliability: Since we have not proved the convergence of the algorithm, we calculate the reliability as the faction of all the cases which converge successfully. Among 1500 cases, only 2 cases can not reach convergence. This shows the algorithm is 99.87% reliable.

3) Speed: Cases were run on a 2.0 GHz Pentium IV Celeron PC with 256 MB of RAM. We specify the speed of the algorithm using two time measurements:

- t_I: Phase I time to perform blocking and starvation analysis.
- t_{II}: Phase II time to perform iterative evaluation.

The average value of t_I is 0.06 second. t_{II} is proportional to the number of iterations and the time per iteration. The average value of t_{II} of the experiment is 15.4 seconds.

VI. Conclusion and Future Research

The purpose of this research is to investigate the behavior of multiple-loop structures and develop an efficient algorithm for evaluating assembly/disassembly networks with arbitrary topologies. By using tools in graph theory and induction method, we are able to efficiently derive the blocking and starvation propagation property of arbitrary complex networks. The algorithm described in this paper provides extremely accurate approximations of expected production rate.

There are several issues will be researched in the future:

- Study the behavior of some typical multiple-loop control structures;
- Formulate some specific optimization problems, e.g., maximize throughput;
- Develop the guideline for designing multiple-loop systems, such as how to design multiple-kanban control structures and loop parameters.

References

[1] A. M. Bonvik. Performance analysis of manufacturing systems under hybrid control policies. Technical Report Ph.D. dissertation, MIT Operations Research Center, 1996.

[2] S. B. Gershwin. An efficient decomposition method for the approximate evaluation of tandem queues with finite storage space and blocking. *Operations Research*, 35(2):291–305, 1987.

[3] S. B. Gershwin. *Manufacturing Systems Engineering*. Prentice-Hall, 1994.

[4] S. B. Gershwin. Design and operation of manufacturing systems: the control-point policy. *IIE Transactions*, 32(10):891–906, 2000.

[5] S. B. Gershwin and L. Werner. An approximate analytical method for evaluating the performance of closed loop flow systems with unreliable machines and finite buffers - part ii: Large loops, 2003.

[6] R. Levantesi. Analysis of multiple loop assembly/disassembly networks. Technical Report Ph.D. dissertation, Politecnico di Milano, 2001.

[7] M. N. S. Swamy. *Graphs, Networks, and Algorithms*. Jon Wiley & Sons, 1981.

[8] T. Tolio, S. B. Gershwin, and A. Matta. Analysis of two-machine lines with multiple failure modes. *IIE Transactions*, 34(1):51–62, 2002.

[9] T. Tolio and A. Matta. A method for performance evaluation of automated flow lines. *Annals of the CIRP*, 47(1):373–376, 1998.

[10] L. Werner. Analysis and design of closed loop manufacturing systems. Technical Report Master's thesis, MIT Operations Research Center, 2001.