JOURNAL OF MATHEMATICAL ANALYSIS AND APPLICATIONS 145, 309–327 (1990)

On the Reversibility of Manufacturing Networks

MAURICE CORTEN AND RENÉ DE KOSTER*,†

Department of Industrial Engineering, Eindhoven University of Technology, P. O. Box 513, 5600 MB Eindhoven, Netherlands

Submitted by E. Stanley Lee

Received April 14, 1988

In this paper it is shown that in many production networks it is possible to reverse the flow direction in one or more buffers without changing the throughput and buffer content distributions of other buffers in the network. If a network possesses the property that simultaneous reversal of the flow direction in all buffers reverses all the buffer contents, then it is called reversible. If it possesses the property that flow reversal in a single buffer b reverses the buffer content of b and leaves all other buffer contents unchanged, then it is called b-reversible. The reversibility of a production network depends, for the discrete product situation, on the blocking rule used. For networks having a continuous product flow the blocking rule is only important for so-called buffersharing networks. b-reversibility is shown to hold for all buffers b for so-called assembly–disassembly networks. Reversibility is shown to hold for buffersharing networks.

1. INTRODUCTION

Recently various authors have paid attention to reversibility of queueing systems with finite buffers. It is observed that reversing the network, that is, reversing the direction of the flow in each buffer, leaves performance measures invariant in certain networks. Muth [6] studies flow lines with finite intermediate buffers and workstations with stochastic service times. He proved that the throughput (mean production rate) of such a line remains unchanged under reversal of flow direction. Yamazaki *et al.* [8] proved that throughput invariance under flow reversal also holds for flow lines with parallel workstations with deterministic service times. For flow lines with single stochastic workstations and finite buffers, Yamazaki *et al.* [7, 8] proved that under flow reversal also the distributions of the time of the nth departure epochs are identical, assuming that the systems are empty initially. Melamed [5] studied single-buffer systems with parallel

* Research supported by the Netherlands Organization for Scientific Research (NWO).

† Present address: Consultants and Engineers Groenewout B.V., P.O. Box 3290, 4300 DG Breda, Netherlands.

0022-247X/90 $3.00

workstations. He identified non-occupied positions in the original network (also called "holes") and some of the occupied positions, with so-called "dual customers," which receive the same service as the ordinary products. Such dual customers appear to correspond to ordinary customers in the reversed line. In all these references the blocking mechanism is the same. If a workstation finishes an operation and the succeeding buffer is full, then the workstation becomes blocked and cannot serve other items.

Ammar [1] and Ammar and Gershwin [2] studied a different blocking mechanism: a workstation does not *start* an operation if the succeeding buffer is full. They considered networks of workstations where assembly/disassembly of products takes place. The layout of these networks is such that a buffer is linked to exactly one upstream and one downstream machine, but a machine may have several upstream or downstream buffers. If a workstation has more than one upstream buffer, it performs assembly of products from the different buffers; if it has more than one downstream buffer, it performs disassembly of the product into disassembly parts. Ammar and Gershwin [2] prove that if the states of two dual networks satisfy so-called duality conditions then the probability of transition between such states in the networks is the same. Two networks are called dual if for each buffer, product motion in the one network corresponds to either product or hole motion in a corresponding buffer in the other network (product motion for buffers with an identical product flow direction and hole motion for buffers with a reversed product flow).

In all references so far the products are discrete. Also continuous product flow models have been studied. De Koster and Wijngaard [4] proved that for certain two-station lines the throughput remains equal under flow reversal.

In this paper both continuous and discrete product flow networks will be discussed. Two types of networks with different layouts are studied. As in Ammar and Gershwin [2] we discuss networks where machines have multiple upstream and downstream buffers, but buffers have only one upstream and downstream machine (assembly/disassembly networks). It is shown that reversing the flow direction in buffer b of the network reverses also the probability distribution of the content of b. That is, if $X(t)$ is the content of b at time t and $X'(t)$ is the content of b at time t in the reversed network, then $X'(t) = K - X(t)$, where K is the capacity of b. This holds at all points in time, provided it is true initially.

The second type of layouts studied are layouts where machines share common upstream or downstream buffers. If an upstream buffer is empty and has a certain (positive) output rate restriction (due to machines supplying the buffer) then the machines obtaining products from that buffer are subject to a priority rule which allocates the speed reduction over the machines. For such networks it is proven that reversing the flow direction

in all buffers of the network, without changing buffer capacities and priority rules, reverses also the probability distributions of the buffer contents. That is, for the content of all buffers j we have $X_j'(t) = K_j - X_j(t)$ at all points in time, provided this is true initially. Quantities denoted with a prime will in the sequel refer to the reversed network. All intermediate buffers in the networks studied in this paper are assumed to have finite capacity.

Note that actually two types of flow reversibility are discussed, namely *reversibility* and *b-reversibility*.

Reversibility of a network means that buffer contents are reversed if the flow in *all* buffers is reversed. A network is *b-reversible* if reversal of the flow direction in buffer b reverses the buffer content of b, whereas the buffer contents of other buffers remain unchanged. If we have b-reversibility for all buffers b then we also have reversibility.

The organization of the paper is as follows. In Section 2 it is shown that continuous flow assembly/disassembly networks are b-reversible for all buffers b. In Section 3 continuous flow networks with buffersharing are studied. It is shown that they are reversible for a variety of priority rules. For a particular priority it is shown in the Appendix that the machine speeds at change points (points in time where full or empty buffers force machines to adapt their speeds) can be determined efficiently. Finally, in Section 4, related results are given for networks with discrete products.

2. Assembly–Disassembly Networks

In this section we consider assembly–disassembly networks, that is, we suppose that machines may have multiple upstream or downstream buffers, but each buffer has at most one upstream machine and at most one downstream machine. In this and the next section it is supposed that the product flow in the network is continuous. An example network is sketched in Fig. 1.

In Fig. 1 buffers are denoted by triangles and machines by rectangles. The network consists of 12 machines and 17 buffers. All buffers, except 1, 5, 10, and 17, are supposed to have finite capacity. The capacity of buffer j is K_j.

The stand-alone (or potential) machine speed of machine i at time t is denoted by $v_i(t)$. The real machine speed, as influenced by the network, is $w_i(t)$. The set of machines is denoted by M, the set of buffers by B, and the set of directed arcs, connecting machines to buffers and vice versa, is denoted by S. The content of buffer i at time t is $X_i(t)$. Buffers that do not have upstream machines are called source buffers. Source buffers are never

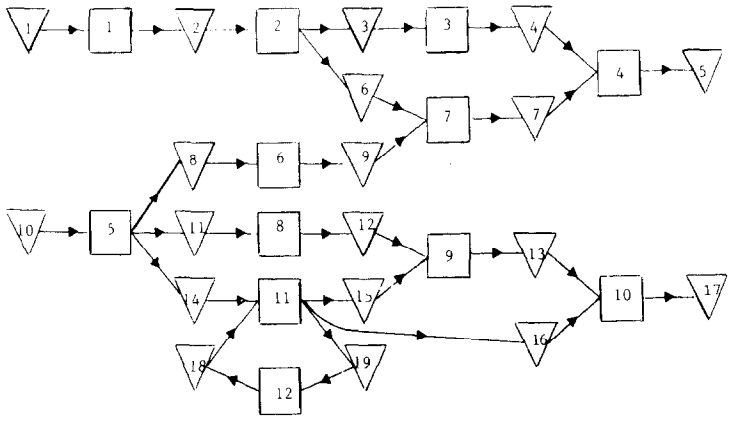

FIG. 1. Assembly–disassembly network.

empty. Buffers that do not have downstream machines are called sink buffers. Sink buffers are never full.

It is assumed that machines that have more than one upstream buffer perform an assembly operation. That is, they assemble one unit of product by taking a fixed amount of product out of each buffer. We suppose one unit of product is taken out of each upstream buffer, but different assembly ratios do not complicate the proof of Theorem 1, to be stated hereafter.

A machine that has more than one downstream buffer disassembles a unit of product into equal amounts of disassembly parts. A machine that performs an assembly and/or a disassembly operation is called an assembly/disassembly machine (a/d machine). Due to the finiteness of the buffer capacities, machines in a/d networks may be slowed down. An a/d machine may be slowed down whenever only one of its upstream buffers is empty or one of its downstreams buffers is full. Suppose, for instance, that in the network of Fig. 1 machine 4 works at speed v_4, machine 3 at speed v_3, and buffer 4 is empty. If $v_3 < v_4$ then machine 4 is slowed down to rate v_3. Since all machines are connected to each other via buffers, slowing down of machines may proliferate through the network.

We say that the slowing down proliferation takes place via so-called *proliferating paths*. A machine is on a proliferating path from machine i, at time t, if this machine can slow down machine i, through a chain of full or empty buffers (if its production rate were smaller than that of machine i). Formally the following definition can be given.

We say that machine j in M is on a *proliferating path from machine i at time t*, if there exists a directed path in the network from machine i to machine j at time t, so that for every buffer on the path it holds that if the

path goes upstream through the buffer, the buffer is empty and if the path goes downstream through the buffer, then it is full. See Fig. 2.

Note that proliferating paths are variable over time. Furthermore they have a direction: machine j in Fig. 2 can slow down machine i, but machine i cannot slow down machine j.

For $i \in M$, $R_i(t)$ is the set of all machines which are on a proliferating path from machine i at time t (including i). Knowledge of $R_i(t)$ is necessary in determining the speed of machine i at time t. In Theorem 1 it is shown that a/d networks are b-reversible for all buffers b. In order to prove this theorem, it is necessary to make some assumptions of the behavior of the machines.

It is assumed that each machine can be in countably many states $\{0, 1, 2, \ldots\}$. The stochastic variable representing the machine state of machine $i \in M$, is denoted by $a_i(t)$. A constant machine speed is associated with each machine state. The sojourn time in each state is a random variable. After a sojourn time in a state, a transition is made to another state. This transition is determined by an irreducible Markov transition matrix. Two types of *change points* are distinguished:

— *change points in machine states*: points in time at which a machine changes its state. It is assumed that these change points do not condense

— *buffer filling or emptying points*: points in time when buffers become full or empty.

It is assumed that these two types of change points do not coincide. Note that the machine speeds in a network only change at such change points.

The aim is now to define the behavior of $a_i(t)$ so that both time-dependent and state-dependent transitions are covered. If the transition rates are time-dependent, they do not depend on the time the machine has been slowed down through full or empty buffers. If a machine has state-dependent transition rates, the transition rates may be influenced by the amount and the time duration of the slow down. As an example, consider an unreliable machine, with production rate 6, (exponential) failure rate

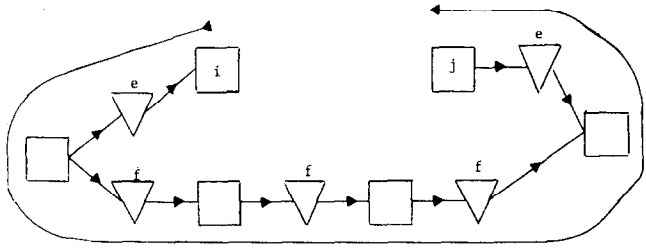

FIG. 2. Proliferating path from machine i to machine j. All buffers are either empty (e) or full (f).

0.1, and (exponential) repair rate. Assume that we are now at time point 0 with the machine up. The failure time is determined by a random draw from an exponential distribution with rate 0.1. Assume that the failure time is 10. Assume furthermore that the machine is slowed down to rate 0 at time 5 and at time 7 it is slowed down to rate 2. Arriving at $t = 5$, due to the memoryless property of the exponential distribution, the new failure time is determined by a random draw from an exponential distribution with rate $0 \times 0.1/6 = 0$. At time $t = 7$, the new failure time is determined by a random draw from an exponential distribution with rate $2 \times 0.1/6$.

More generally, let the state of machine i at change point t be $\{a_i(t)\}$. The sojourn time in that state is determined as follows. If the machine is not slowed down at time t, then the sojourn time is determined by a random draw from the same distribution as if the machine were in stand alone position. If the machine *is* slowed down at time t, then the sojourn time is determined by a random draw from a distribution that may depend on the amount and duration of slow down. In all cases, the sojourn time distribution is assumed to have a positive mean.

(2.1)

These change points generated by a random draw from a certain distribution are denoted by *projected change points*, since these generated change points are not necessarily realized. The *stand-alone* (or *potential*) machine speed, corresponding to the realization of the machine state $\{(t, a_i(t));\ t \geqslant 0\}$, is denoted by $v_i(t)$. The real machine speed of machine i, as influenced by the network, is denoted by $w_i(t)$. This machine speed is given by

$$w_i(t) = \min\{v_j(t);\ j \in R_i(t)\}. \tag{2.2}$$

THEOREM 1 (b-Reversibility of a/d networks). *Let N be an a/d network, with machines modeled as above, with sojourn time distributions in each state as defined above. Let $\ell \in B$, the set of buffers in N, and let N' be the network arising from N by reversing the direction of the flow in buffer ℓ. If this results in a machine without upstream buffer, then add a source buffer supplying this machine. If the flow reversal results in a machine without downstream buffer, then add a sink buffer obtaining parts from that machine. The cardinality of M is denoted by n, and the cardinality of B by m. Let $\{(t, a_1(t), ..., a_n(t));\ t \geqslant 0\}$ be a stochastic realization of the machine states in N, $\{(t, X_1(t), ..., X_m(t));\ t \geqslant 0\}$ the corresponding realization of the buffer contents, and $\{(t, w_1(t), ..., w_n(t));\ t \geqslant 0\}$ the corresponding realization of the machine speeds in N. Define*

$$\{X_j'(0)\} = \{X_j(0)\}, \qquad j \in B, \quad j \neq \ell, \qquad \{X_\ell'(0)\} = \{K_\ell - X_\ell(0)\}$$
$$\{a_i'(0)\} = \{a_i(0)\}, \qquad i \in M.$$

Then $\{(t, a_1(t), ..., a_n(t));\ t \geqslant 0\}$ *is a stochastic realization of the machine states in* N', $\{(t, X_1(t), ..., K_\ell - X_\ell(t), ..., X_m(t));\ t \geqslant 0\}$ *the corresponding realization of the buffer contents, and* $\{(t, w_1(t), ..., w_n(t));\ t \geqslant 0\}$ *the corresponding realization of the machine speeds* N'.

Proof. Note first that if at time t we have $\{a_i'(t)\} = \{a_i(t)\}$ for all $i \in M$ and $X_j'(t) = X_j(t)$, for all $j \neq \ell$, $X_\ell'(t) = K_\ell - X_\ell(t)$, then for all $i \in M$, the set of machines that are on a proliferating path from machine i at time t is the same in N and N'. Note that the flow direction through buffer ℓ is reversed, but simultaneously $X_\ell'(t) = K_\ell - X_\ell(t)$ and hence if machine j slows down machine i by a proliferating path through buffer ℓ in N, then it also slows down machine i in N'. This means that

$$R_i'(t) = R_i(t) \text{ and hence, by (2.2), } w_i'(t) = w_i(t), \qquad \text{for all } i \in M. \tag{2.3}$$

The proof is now by induction to change points in the machine states. By assumption, the theorem holds for $t = 0$ (note that $R_i'(0) = R_i(0)$, for all $i \in M$, see (2.3)).

Assume the assertions of the theorem hold until the change point in the state of machine i, t, say. At time t, the projected change point t_2 in the state of machine i is the same in N and N', since in both networks the machine states are equal and, in case of slow down of machine i, the amount and duration of slow down of machine i are also equal. Hence $t_2' = t_2$. Let t_1 be the next change point, either in a machine state or in a buffer content, in N. There are now two possibilities.

1. t_1 is a change point in the state of a machine j, say. Since all change points, machine states, and machine speeds are the same in N and N', prior to t, by the induction hypothesis, it follows by (2.1) that all projected change points in machine states are also the same in N and N'. Hence, $t_1' = t_1$ and the machine states are identical in N and N' between t and t_1. Since all machine speeds are equal in N and N' at time t and since they do not change until t_1, it follows that $X_j'(t_1) = X_j(t_1)$, for all $j \neq \ell$, and $X_\ell'(t_1) = K_\ell - X_\ell(t_1)$. By (2.3) it now follows that also $w_i'(t_1) = w_i(t_1)$, for all $i \in M$.

2. t_1 is a change point in the state of buffer j. Similarly as under (1), it follows by the induction hypothesis that $t_1' = t_1$ and $\{a_i'(t_1)\} = \{a_i(t_1)\}$ for all $i \in M$. Assume that j becomes empty at t_1 in N. Since the machine speeds are equal in N and N' at time t and since the machine speeds do not change between t and t_1, it follows that j also becomes empty at $t_1' = t_1$ in N', if $j \neq \ell$. If $j = \ell$, then j becomes full at t_1 in N'. Hence $X_r'(t_1) = X_r(t_1)$, for all $r \neq \ell$, $X_\ell'(t_1) = K_\ell - X_\ell(t_1)$. By (2.3) it now follows that $w_i'(t_1) = w_i(t_1)$, for all $i \in M$. Hence, the assertions of the theorem hold at time t_1. At time t_1, new projected change points in the states of the slowed

down machines are generated in N and N'. By (2.1) it follows that these change points coincide in N and N'. We can now take the next change point t_3 and show in a similar way that the assertions of the theorem hold at t_3. By assumption, the change points do not condense. We can proceed in this way, until we arrive at a change point in a machine state.

In both cases, it follows that the assertions of the theorem hold at the next following change point in a machine state. This concludes the proof.

Theorem 1 states that a/d networks are b-reversible, for all $b \in B$. If a unique equilibrium distribution for the buffer contents exists for an a/d network N, then it is the same or reversed for every network N' in which the flow through one or more buffers is reversed. That is, if N' arises from N by reversing the flow in buffer ℓ, then

$$P[X'_j \leqslant x] = P[X_j \leqslant x], \quad \text{for } j \neq \ell, \quad \text{and} \quad P[X'_\ell \leqslant x] = P[X_\ell \geqslant K_\ell - x].$$

Note that even when there exists an equilibrium distribution it need not be unique. It may depend on the initial state of the system. In the network of Fig. 3, machine 1 performs disassembly and machine 2 performs assembly. The initial state $X_1(0) = K_1$, $X_2(0) = 0$ in this figure leads to

$$\lim_{t \to \infty} X_1(t) = K_1, \qquad \lim_{t \to \infty} X_2(t) = 0.$$

The initial state $X_1(0) = 0$, $X_2(0) = K_2$ leads to $\lim_{t \to \infty} X_1(t) = 0$, $\lim_{t \to \infty} X_2(t) = K_2$.

It is clear that networks as in Fig. 3 are only more or less realistic if $K_1 = K_2 =: K$.[1] However, whenever this is the case the system of Fig. 3 is equivalent to an ordinary flow line containing only one intermediate buffer of capacity K.

By Theorem 1, we can replace directed cycles in a/d networks by bypasses. If the network contains then bypasses as in Fig. 3 (that is, without intermediate machines within the bypass), these can be eliminated too, under the conditions mentioned before. This fact can be used in approximation algorithms for a/d networks. See De Koster [3].

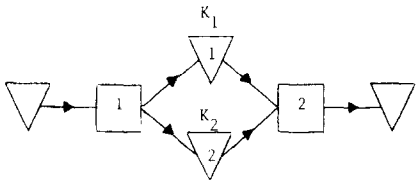

FIG. 3. Two-machine a/d network.

[1] And if we start with $X_1(0) = X_2(0)$.

In Ammar and Gershwin [2], a/d networks that can be obtained from each other by reversing the flow direction in one or more buffers (and adding source and sink buffers if necessary) are called equivalent. They show how the equivalence class of a tree-shaped a/d network can be constructed.

3. BUFFERSHARING NETWORKS

In this section we will consider continuous flow networks of linked buffers and machines in which several machines may obtain products from a single buffer and may supply to a single buffer. However, each machine has only one upstream and one downstream buffer. Such a network is called a buffersharing network. Since several machines may share a common finite capacity buffer, we have to do with a global restriction, holding for all machines, rather than a local restriction, holding for each machine separately. This implies that when several machines obtain from a single buffer and that buffer is (nearly) empty, then it has to be decided which machines are allowed to operate on the products and with which rates (in the continuous flow situation). A similar situation arises when several machines supply a single buffer. In general, the installation of such a common buffer, preceding several machines, improves the performance of the system. If products are allocated to a single machine and that machine fails, then the system stops, whereas if the products are not allocated beforehand, they can be processed by other machines. A complex example of a buffersharing network is sketched in Fig. 4.

In the buffersharing networks studied in this section it is assumed that there is a (directed) path from a source buffer to a sink buffer. As in a/d

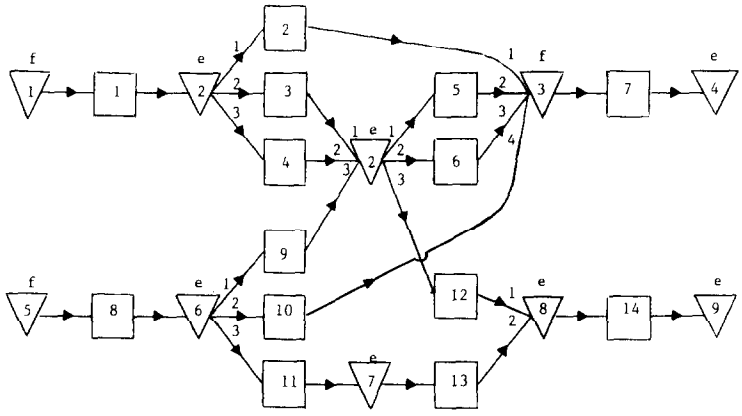

FIG. 4. Buffersharing network.

networks, machines may be forced to slow down if a (directly) preceding buffer is empty or a (directly) succeeding buffer is full. Suppose buffer $b \in B$ is full and let it be preceded by machines $n_1, ..., n_r$ and succeeded by machines $m_1, ..., m_s$. Let the potential speed of machine n_i be denoted by $v(n_i)$. If the total potential speed of all machines $n_1, ..., n_r$ (say $\sigma_1 := \sum_{i=1}^{r} v(n_i)$) is greater than the total potential speed of machines $m_1, ..., m_s$ (say σ_2) at that moment, then machines $n_1, ..., n_r$ will have to slow down till their total speed is less than or equal to the total speed of machines $m_1, ..., m_s$. The reduction in the potential speeds of the machines $n_1, ..., n_r$ due to this is determined by a priority rule. With each possible priority rule there is associated a system of equations determining the machine speeds.

An example of such a priority speed-allocation rule is the following priority rule 1:

R_1: Allocate the necessary reduction in potential speeds to the machines $n_1, ..., n_r$ in order of decreasing priority. The priority of the arc (n_i, b) is greater than the priority of (n_j, b) iff $n_i < n_j$. The speed of machine n_i, $w(n_i)$, now becomes $\min\{v(n_i), \sigma_2 - \sum_{n_j < n_i} w(n_j)\}$, where σ_1 and σ_2 are defined as above. If $\sigma_2 > \sigma_1$ and the buffer is empty, then the machine speeds are determined similarly.

Note that this priority speed-allocation rule is locally defined. In buffersharing networks, the allocation rule must be applied at each full or empty buffer. This leads, for all machines, to a number of equations that their speed must satisfy. For the network sketched in Fig. 5 this rule results in the following speeds: $w(1) = \min\{4, 2\} = 2$, $w(2) = \min\{1, w(1), 2\} = 1$, $w(3) = \min\{3, 2 - w(1), 2 - w(2)\} = 0$.

Another priority rule is the following:

R_2: Allocate the reduction in potential speeds of the machines $n_1, ..., n_r$ to these machines equally. The speed of machine n_i then becomes

$$w(n_i) = \frac{v(n_i)}{\sum_{j=1}^{r} v(n_j)} \cdot \sigma_2.$$

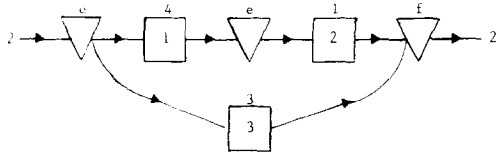

FIG. 5. Four-machine network. On top of machine i, $v(i)$ is indicated. For a buffer, e denotes that the buffer is empty, f denotes that the buffer is full.

This priority rule results in the following speeds for the network in Fig. 5: $w(1) = \frac{8}{7}$, $w(2) = \min\{w(1), \frac{1}{2}\} = \frac{1}{2}$, $w(3) = \min\{\frac{6}{7}, \frac{6}{4}\} = \frac{6}{7}$.

Many other priority rules are possible. However, it may be possible that they lead to contradictory systems of equations for the machine speeds or systems with multiple solutions. In the Appendix it is shown that for networks with priority rule R_1, with some extra restrictions, the system of equations for the machine speeds is solvable at all points in time with a unique solution. An efficient algorithm based on dynamic programming is given for obtaining the machine speeds.

Now a theorem is presented for mixtures of buffersharing and a/d networks, analogous to Theorem 1. A machine that has multiple upstream or downstream buffers in such a network performs assembly or disassembly, respectively. For each buffer with multiple upstream and downstream machines an allocation rule is needed. We do not give allocation rules for such networks, since only plain a/d and plain buffersharing networks are studied in the rest of this text. However, it is not difficult to generalize rules R_1 and R_2 for such combined networks.

Let N be a mixture of a buffersharing and an a/d network and let N' be the network arising from N by reversing the flow direction of buffer $\ell \in B$. If this results in a machine without upstream buffer then this machine becomes a disassembly machine and a source buffer is added supplying this machine. If the flow reversal results in a machine without downstream buffer then this machine becomes an assembly machine supplying a sink buffer. Hence, the resulting network is again a mixture of a buffersharing and an assembly–disassembly network.

Starting from a plain buffersharing network, flow reversal only results in another buffersharing network, if the flows in *all* buffers are reversed. The speed allocation rule in mixtures of buffersharing and a/d networks must satisfy certain conditions.

Let N and N' be networks as above. A priority speed-allocation rule R in a network N, which is a mixture of a buffersharing and an a/d network, is said to possess the *local reversibility property* if

1. for all realizations of machine states $\{(t, a_i(t)) \mid t \in \mathbb{R}_+, i \in M\}$ and for potential machine speeds $\{(t, v_i(t)) \mid t \in \mathbb{R}_+, i \in M\}$, the used allocation rule R results in a *unique* set of real machine speeds $\{w_i(t) \mid t \in \mathbb{R}_+, i \in M\}$.

2. If at an arbitrary time t, the realization $\{a_i'(t)\} = \{a_i(t)\}$, for all i and $X_j'(t) = X_j(t)$, $j \neq \ell$, $X_\ell'(t) = K_\ell - X_\ell(t)$, then allocation rule R determines the real machine speeds so that $w_i'(t) = w_i(t)$, for all i.

Note that for the case where N and N' are plain buffersharing networks, both R_1 and R_2 possess the local reversibility property.

For mixture networks N and N' as defined above, we assume that

— the assumptions for the machine states are the same as those for a/d networks.

— The used allocation rule R possesses the local reversibility property.

THEOREM 2 (b-Reversibility of mixed buffersharing a/d networks). *Let* $\{(t, a_1(t), ..., a_n(t));\ t \geqslant 0\}$ *be a stochastic realization of the machine states in* N, $\{(t, X_1(t), ..., X_m(t));\ t \geqslant 0\}$ *the corresponding realization of the buffer contents, and* $\{(t, w_1(t), ..., w_n(t));\ t \geqslant 0\}$ *the corresponding realization of the machine speeds in* N. *Define*

$$\{X'_j(0)\} = \{X_j(0)\}, \qquad j \in B, \quad j \neq \ell, \qquad \{X'_\ell(0)\} = \{K_\ell - X_\ell(0)\}$$
$$\{a'_i(0)\} = \{a_i(0)\}, \qquad i \in M.$$

Then $\{(t, a_1(t), ..., a_n(t));\ t \geqslant 0\}$ *is a stochastic realization of the machine states in* N', $\{(t, X_1(t), ..., K_\ell - X_\ell(t), ..., X_m(t));\ t \geqslant 0\}$ *the corresponding realization of the buffer contents, and* $\{(t, w_1(t), ..., w_n(t));\ t \geqslant 0\}$ *the corresponding realization of the machine speeds* N'.

Sketch of the Proof. The proof is along the same line as that of Theorem 1. Suppose the assertions of the theorem hold until the change point in a machine state t. Let t_1 be the next change point, either in machine states or in buffer contents, in N. As in the proof of Theorem 1, there are two possibilities, of which only one is treated:

Assume that t_1 is a change point in the state of buffer j. Assume, without loss of generality, that j becomes empty at t_1. Similarly as in the proof of Theorem 1, it follows by the induction hypothesis that $t'_1 = t_1$ and $\{a'_i(t_1)\} = \{a_i(t_1)\}$ for all $i \in M$. Since the real machine speeds in N and N' are equal at t, and since they do not change until t_1, it follows that j also becomes empty at t_1 in N', if $j \neq \ell$ and j becomes full at t_1 in N', if $j = \ell$. Hence $X'_j(t_1) = X_j(t_1)$, for all $j \neq \ell$, $X'_\ell(t_1) = K_\ell - X_\ell(t_1)$. By the local reversibility property of the allocation rule R, it follows that $w'_i(t_1) = w_i(t_1)$, for all $i \in M$. Hence, the assertions of the theorem hold at time t_1. We can now take the next change point t_2 and show in a similar way that the assertions of the theorem hold at t_2. By assumption, the change points do not condense. We can proceed in this way until we arrive at a change point in a machine state. This concludes the proof.

Note also that directed cycles or bypasses in the network are no problem in the proof of Theorem 2.

4. Networks with Discrete Products

Till now we assumed that the product flow in the networks was continuous. It is also possible to derive reversibility results for a/d networks with discrete products and general service mechanisms, as long as the machines do not have storage capacity. That is, we suppose that a machine does not start an operation as long as some immediately downstream destination buffer is full, and on production completion a product is immediately transferred from upstream buffer to downstream buffer. The non-occupied positions, or holes, move simultaneously with the ordinary products through the network, but in opposite direction. For such a blocking mechanism it is easy to see that if the flow direction in buffer ℓ is reversed, then hole motion in buffer ℓ in the original network corresponds to product motion in ℓ in network N'. It is possible to prove a theorem similar to Theorem 1 (for a/d networks) for the discrete product case. This can be done by comparing the state of the machines in the a/d networks N and N' at points in time where a machine has finished a product and just stored it in a buffer.

Let $s_i(t)$ be the service time of machine $i \in M$ starting service at time t. This service time may depend on the realised service times in the past, but is independent from the upstream and downstream buffer contents. $t_0 = 0, t_1, t_2, \ldots$ are points in time where either a machine finishes a product or starts servicing a new product. Let

$e(t) :=$ the set of machines servicing or starting service at time t,

$u(t) :=$ the set of machines waiting or finishing a product at time t.

The intersection of the sets $e(t)$ and $u(t)$ is not necessarily empty. For $i \in e(t_m)$,

$$l_i(t_m) := \text{the latest point in the time interval } [0, t_m], \text{ machine } i \text{ started service.}$$

Suppose we have $t'_0 = t_0$, $e'(0) = e(0)$, $u'(0) = u(0)$, $X'_j(0) = X_j(0)$, $j \neq \ell$, and $X'_\ell(0) = K_\ell - X_\ell(0)$. Then we have

Theorem 3. *For each realization of the machine speeds*:

1. $t'_m = t_{m'}$ *for* $m \geqslant 0$.
2. $e'(t) = e(t)$, $u'(t) = u(t)$, $l'_i(t) = l_i(t)$ *for all* $t \geqslant 0$.
3. $X'_j(t) = X_j(t)$, *for all* $j \in B$, $j \neq \ell$, $t \geqslant 0$, $X'_\ell(t) = K_\ell - X_\ell(t)$, *for all* $t \geqslant 0$.

Sketch of the Proof. By induction to m. Suppose the theorem holds for all values $\leqslant m$. The machines $i \in e(t_m) = e'(t_m)$ started servicing at time $l_i(t_m) = l_i'(t_m)$. Furthermore $e'(t_m) = e(t_m)$ and $u'(t_m) = u(t_m)$ (induction) and hence $s_i(l_i(t_m)) = s_i'(l_i'(t_m'))$.

Hence $t'_{m+1} = t_{m+1} = \min\{l_i(t_m) + s_i(l_i(t_m)); i \in e(t_m)\}$. Between t_m and t_{m+1}, $e(t)$, $u(t)$, and $X_j(t)$ do not change. At t_{m+1} in both N and N' the same machines have finished their product. Therefore all buffer transitions are equal except in ℓ, where the buffer transition is reversed, and the same machines can start servicing a product at t_{m+1}, which implies $e'(t_{m+1}) = e(t_{m+1})$, $u'(t_{m+1}) = u(t_{m+1})$.

It is also possible to prove a theorem analogous to Theorem 2 for mixed buffersharing a/d networks. In defining a priority rule for machines obtaining from the same buffer or supplying the same buffer, we again have to suppose that machines do not have capacity for products and furthermore a machine being busy or idle at a certain point in time has to be in concordance with being busy or idle in the reversed network. For example, if n machines $i_1, \ldots, i_n$ have a common upstream buffer b in the network N, then if machine i_r is working with b having content x (integer), then i_r must also be working in N' with b having content $K_b - x$.

An example of a priority rule which leads to reversible networks is priority rule R_3, which is explained by means of an example network. In this example network, four machines (machines 1, 2, 3, and 4) share a common upstream buffer b of capacity 5.

R_3: Machine i has priority over machine j iff $i < j$. Hence if the content of b is 3, then machines 1, 2, and 3 are busy (if not blocked) and machine 4 is starved.

A similar statement can be made if these four machines would share a common downstream buffer b of capacity 5. If then the content of b is 3, then machines 1 and 2 are busy (if not idle) and machines 3 and 4 are blocked.

These priorities are preemptive, that is, for the case of the common downstream buffer, if the content of b is 3 and machine 1 finishes an operation first, then machine 2 is preempted. This is possible since machine 2 has no capacity for products and the part was not taken out of the upstream buffer of machine 2. Whenever machine 2 starts with a new operation it starts from the beginning.

For priority rule R_3 it is possible to prove a theorem similar to Theorem 3, but now for mixed buffersharing a/d networks.

The line of proof followed in the previous theorems cannot be followed for the so-called transfer type blocking mechanism as used by Muth [6] (this blocking mechanism is often called type 1 blocking). For the blocking

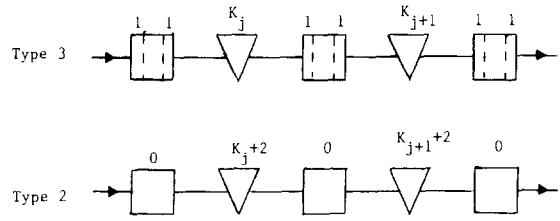

FIG. 6. Flow lines with type 3 and type 2 blocking.

mechanism of this paper (often denoted by communication type blocking, or type 2 blocking) there is complete correspondence between part movement in N and part or hole movement in N'. Parts in a buffer b in N correspond to parts in this buffer in N' if the flow direction through b is not reversed. Parts in a buffer b in N correspond to holes in N', if the flow direction through b is reversed. For type 1 blocking there is no such correspondence. However, for the following transfer type blocking mechanism (denoted by blocking type 3) the correspondence between product movement and hole movement still exists.

BLOCKING TYPE 3. A machine has capacity for one product, when not blocked. The operation takes place in the machine. A machine starts an (assembly) operation as soon as there are upstream products in each direct upstream buffer. If there is no storage room for the finished product(s), then the machine becomes blocked. The machine has in blocked state the storage capacity for an extra product.

A flow line with type 3 blocking with buffer capacities K_j corresponds to the same line with buffer capacities $K_j + 2$, but with type 2 blocking. See Fig. 6.

In both lines of Fig. 6 the capacities are indicated. Let the two extra buffer positions of the type 2 blocking mechanism correspond with the machine-capacity positions of the type 3 blocking mechanism. See Fig. 7. For both blocking types each machine can be in three different states, namely busy, blocked, and starved. The distribution of the products over the line for these machine states is the same for both blocking mechanisms and is indicated in Fig. 8.

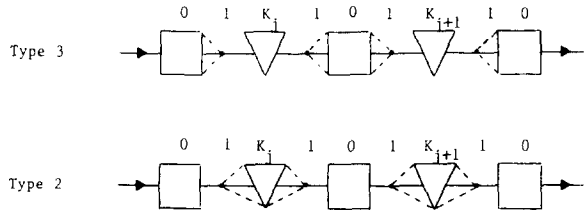

FIG. 7. Product positions for both blocking mechanisms.

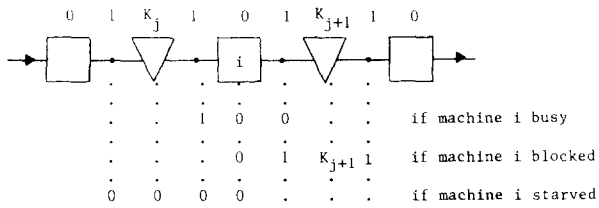

FIG. 8. Product positions for both type 2 and type 3 blocking. A '0' means that a position is not occupied. A '1' means that a position is occupied and a dot represents either an occupied or a non-occupied position.

APPENDIX

In this appendix it will be illustrated how the machine speeds can be determined in continuous flow buffersharing networks. The used priority rule is rule R_1, defined in Section 3. This priority rule determines the machine speeds in buffersharing networks uniquely, if no loops or bypasses occur. If a network contains loops or bypasses the system of equations for the machine speeds may be contradictory or have multiple solutions. Suppose that machine 3 in the network in Fig. 5, which contains a bypass, has priority over machine 1. Machine 2 has priority over machine 3. In this case the system of equations for the machine speeds, using R_1, is

$$w(1) = \min\{4, 2 - w(3)\}$$
$$w(2) = \min\{1, 2, w(1)\}$$
$$w(3) = \min\{3, 2, 2 - w(2)\}.$$

This leads to $w(2) = \min\{1, 2 - \min\{2, 2 - w(2)\}\} = \min\{1, \max\{0, w(2)\}\}$. Hence, for each $w(2) \in [0, 1]$, there is a solution of this system of equations. In order to avoid such situations it is assumed that the priority rule satisfies the following additional conditions.

(1) The network contains no loops (directed cycles).

(2) For every bypass (a set of two different paths from a buffer $b_1 \in B$ to another buffer $b_2 \in B$) that the network contains, it holds that if the priority of the first arc of the one path is greater than the priority of the first arc of the other path at b_1, then the same has to hold at b_2 for the last arc of both paths.

In order to determine the machine speeds the following additional variables are needed. With each buffer b_1 that has downstream machines there is associated an output rate restriction $o(b_1)$, which is the maximum possible output rate of products from this buffer. If b_1 is non-empty then

$o(b_1) = \infty$. This output rate restriction means that preceding machines may be blocked. With each buffer b_2 that has upstream machines there is associated an input rate restriction $i(b_2)$, which is the maximum possible input rate into this buffer. If b_2 is non-full then $i(b_2) = \infty$. The input rate restriction means that following machines may be starved.

It is assumed that the network is connected. If not, each connected component can be treated individually.

An algorithm is described that determines the machine speeds for each combination of buffer contents (either empty, full, or non-empty and non-full). The algorithm locates a machine of which the speed can be calculated and removes this machine with every connecting arc from the network. Non-connected buffers are also removed. The input and output rate restrictions of the remaining buffers are adapted, depending on the speed of this particular machine. This procedure is repeated till all machine speeds have been determined. At each removal one or more networks remain which are of the same types as the original network. Let B denote the set of buffers, $v(i)$ the potential machine speed of machine i, and $w(i)$ the real speed of machine i as influenced by the network. pred_i is the buffer preceding i and succ_i is the buffer succeeding i.

Algorithm
initialization for all full buffers b: $i(b) := 0$;
for all empty buffers b: $o(b) := 0$;
while $B \neq 0$
do found := false;
choose source buffer a;
while not found
do while a is not a sinkbuffer **and not** found
do go downstream along the arc with the highest priority
to machine i and further downstream to buffer b;
if b is not full
then found := true;
$w(i) := \min\{v(i), o(a)\}$
else $a := b$
fi
od;
if a has no other incoming arcs with a higher priority
than the arc along which a was entered **and not** found
then found := true;
$w(i) := \min\{v(i), i(a), o(\text{pred}_i)\}$
fi;
while a is not a source buffer **and not** found
do go upstream along the arc with the highest priority
to machine i and further to buffer b
if b is not empty
then found := true;
$w(i) := \min\{v(i), i(a)\}$

```
            else a := b
            fi
        od;
        if a has no other outgoing arcs with a higher priority
          than the arc along which a was entered and not found
        then found := true;
             w(i) := min{v(i), o(a), i(succ_i)}
        fi;
    od;
    remove i from the network with all its arcs and non-connected
    buffers;
    if i obtained from an empty buffer b
    then o(b) := o(b) - w(i) fi;
    if i supplied an empty buffer b
    then o(b) := o(b) + w(i) fi;
    if i obtained from a full buffer b
    then i(b) := i(b) + w(i) fi;
    if i supplied a full buffer b
    then i(b) := i(b) - w(i)
    fi
od.
```

Conditions (1) and (2) guarantee that the algorithm always takes a new machine and a new buffer while going downstream or upstream through the network. The finiteness of the network guarantees that a machine can always be found of which the speed can be determined. If the remaining network is disconnected then the components are treated one by one. For the production system sketched in Fig. A1 the algorithm gives the following results, in case buffers b_1 and b_3 are full and buffer b_2 is empty. Denote the source buffer (preceding machine m_1) by b_0, and the sink buffer (succeeding machine m_6) by b_4. At each buffer shared by more than one machine, machine m_i has priority over machine m_j iff $i < j$.

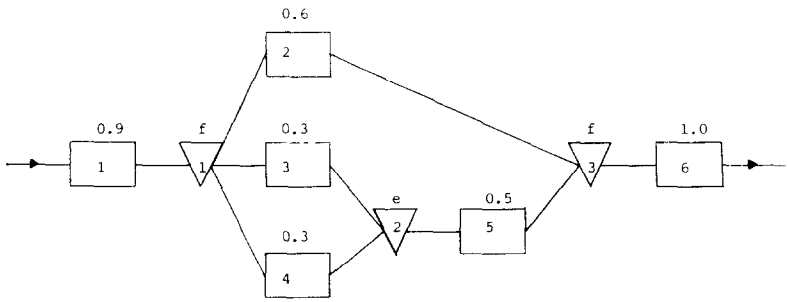

FIG. A1. Buffersharing network. On top of the machines the speeds are indicated. All buffers are either full (f) or empty (e).

Initialization.

$$o(b_0) = i(b_4) = o(b_1) = i(b_2) = o(b_3) = \infty,$$
$$i(b_1) = o(b_2) = i(b_3) = 0.$$

Machine Speed Determination. $w(m_6) = 1$; $i(b_3) = 1$; $w(m_2) = 0.6$; $i(b_1) = 0.6$; $i(b_3) = 0.4$; $w(m_3) = 0.3$; $o(b_2) = 0.3$; $i(b_1) = 0.9$; $w(m_4) = 0.3$; $o(b_2) = 0.6$; $i(b_1) = 1.2$; now we have two components left, namely the systems $b_0 - m_1 - b_1$ and $b_2 - m_5 - b_3$.

$w(m_1) = 0.9$; $i(b_1) = 0.3$; $w(m_5) = 0.4$.

REFERENCES

1. M. H. AMMAR, "Modeling and Analysis of Unreliable Manufacturing Assembly Networks with Finite Storages," MIT Laboratory for Information and Decision Systems, Report LIDS-TH-1004, 1980.
2. M. H. AMMAR AND S. B. GERSHWIN, Equivalence relations in queueing models of manufacturing, in "Proceedings, Nineteenth IEEE Conference on Decision and Control, 1980," pp. 715–721.
3. M. B. M. DE KOSTER, "Approximation of Assembly–Disassembly Systems," Report BDK/ORS/87-01, Eindhoven University of Technology, 1987.
4. M. B. M. DE KOSTER AND J. WIJNGAARD, On the equivalence of multistage production lines and two-stages lines, *IIE Trans.* **19** (1987), 351–354.
5. B. MELAMED, A note on the reversibility and duality of some tandem blocking queueing system, *Management Sci.* **32** (1986), 1648–1650.
6. E. J. MUTH, The reversibility property of production lines, *Management Sci.* **25** (1979), 152–158.
7. G. YAMAZAKI, T. KAWASHIMA, AND H. SAKASEGAWA, Reversibility of tandem blocking queueing systems, *Management Sci.* **31** (1985), 78–83.
8. G. YAMAZAKI, H. SAKASEGAWA, AND T. KAWASHIMA, Production rate estimated with flow-shop reversibility, *Bull. Japan Soc. Mech. Eng.* **21** (1978), 167–171.