Improving efficiency of decomposition methods for closed-loop continuous-flow production models

David Douard [a,1] and Bruno Baynat [b]

[a] ***douard@lgi.ecp.fr***
Laboratoire de Génie Industriel, Ecole Centrale de Paris
Grande Voie des Vignes, 92295 Châtenay-Malabry, France

[b] ***Bruno.Baynat@lip6.fr***
Laboratoire d'Informatique de Paris 6, Université Pierre et Marie Curie
8 rue du Capitaine Scott, 75015 Paris, France

Abstract

This paper presents a new analytical method for estimating the performance parameters of a closed-loop production line with unreliable machines and finite buffers. Even though there have been many studies on the closed-loop queueing networks, where servers have stochastic service time but are reliable (see eg. [1,13–15]), few practical works have been made concerning the production line paradigm [10,6]. The presented technique is based on the same decomposition principle than the original method proposed in [9]. The objective is to decompose the production line into a set of two-machine lines, and to reduce the analysis of the initial line to one of these simpler systems that can be analyzed with exact techniques. The new method drastically reduces the complexity of the underlying algorithm by modifying the system of equations (but not its solution), and by eliminating the double-loop structure and the dissymmetry of the initial algorithm. Moreover the new technique turns out to be much more robust and and has proven to converge quickly in almost all situations, including examples with a very high variability in parameters, whereas the original algorithm fails to convergence in many examples, including practical cases. This objective of this paper is also to propose a unified framework in order to gather together the different techniques and to prepare the extension of the resolution to manufacturing systems with much more general topology (including routing and Fork/Join mechanisms).

Key words: Production, Production line, Closed-loop, Decomposition

[1] To whom correspondence should be adressed.

Preprint submitted to Elsevier Science *13 November 2003*

Introduction

This paper is concerned with a class of production systems referred to as *closed-loop production lines* [6,9]. Closed-loop queuing networks with finite capacity buffers have been studied in several papers, but have a significantly different assumptions on the behaviour of the system, compared to the production systems which are based on the models from [10]. Namely, in queuing network systems, machines are reliables but have a stochatsic service time, whereas in production systems, machines have constant service times, but are unreliable. See [6] for a comparison between these two approaches, and eg. [1,13–15], concerning the closed-loop queuing network paradigm.

In production line systems, when raw parts enter the system they are loaded onto pallets at a loading station. Then they visit a fixed sequence of processing machines, intermediately wait in buffer spaces, and finally leave the system at an unloading station. As soon as a finished product is unloaded, the free pallet that was attached to it is sent back to the loading station. We assume that there are always raw parts available at the input and that there is always room for finished products at the output. Consequently, we can focus solely on the circulation of pallets, that can be considered as the only non-consumable resource for the production system. Machines are subject to failures and we make the classical assumption [2,6,3] that the failures are operation dependent (meaning that a machine can fail only when it is working). Intermediate buffers in between machines reduce the effects of failures of a machine onto the others.

In order to obtain the performance parameters of manufacturing systems, such as the production rate or the average amount of material in each buffer, an alternative to simulation is to use analytical techniques. Most of the techniques that have been proposed are based on decomposition. Decomposition techniques have initially been developed for *"open" production lines* with unreliable machines and finite buffers [10,4,5]. The principle of these techniques is to decompose the system into a set of smaller subsystems consisting of two-machine lines, and to relate the parameters of the original line to the one of the smaller subsystems.

The extension of these techniques to closed-loop production lines is not trivial. Indeed, the population constraint involved by the fixed number of pallets in the system makes the behavior of closed-loop production lines more complex and the techniques to solve them more complicated. A first analytical method for evaluating closed-loop production systems was proposed in [9] and was an extension of the initial technique [4,5] for open lines. The principle was again to decompose the original line into a set of two-machine lines. The system of equations derived from this decomposition is basically the same as the one originally obtained in [4,5] except that one equation of this system is now removed and a new one that explicitly expresses the population constraint is added. To solve this new system [9] proposes an algorithm based on the principle of the dichotomy and that involves the original algorithm [4,5] as a building block (subfunction). The resulting algorithm has thus two levels of iterations.

In this paper we propose a new technique to solve the same system of equations, but in a single-loop process. The idea is to modify the writing of the system, without modifying its solution, by imposing a constraint on a whole set of parameters. The objective is both to eliminate the double-loop structure of the previous algorithms and the dissymmetry of the process related to the choice of a given parameter (to make the dichotomy), in order to give a more efficient and a more robust technique.

The goals of this paper are:

- to formalize the problem and to propose a unified framework to gather together the different techniques;
- to compare the different techniques and understand how they differ from each other;
- to show that the previous techniques ([9] and its unpublished extension) have convergence problems that prevent the algorithms from converging in many examples, including practical cases;
- to understand where these problems come from;
- to propose a new efficient and robust method for solving the same system of equation, and compare it to the previous techniques in terms of efficiency and robustness;
- to prepare the extension of the resolution to manufacturing systems with much more general topology (including routing and Fork/Join mechanisms).

Finally, we must note that recent works have been developped for solving the same closed-loop production line models as the one studied in this paper. These techniques are again extensions of the original decomposition method initially presented in [10] for open lines. The first extension [11] proposes to take into account the correlation between number of parts in the different buffers. However the model is much more complex. Attemps to reduce the complexity of this extension are presented in [12].

The paper is organized as follows. Section 1 introduces the closed-loop production line model. The decomposition principle and the formalization of the system of equations are developed in Section 2. The different solution methods are presented in Section 3. Finally, numerical experiments are reported in Section 4.

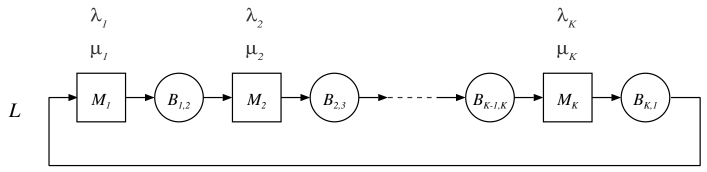

Fig. 1. Production line model.

1 The model

1.1 *The continuous-flow asynchronous model*

We consider a closed-loop production line consisting of K machines $\{M_i\}_{i\in\{1..K\}}$ separated by K intermediate buffers $\{B_{i,i+1}\}_{i\in\{1..K\}}$ as illustrated in Figure 1. Parts are carried by a fixed number of N pallets throughout the production line. The line is supposed to be homogenous, thus all the machines have the same constant processing time denoted by T. Without loss of generality, we can assume that $T = 1.0$, as the time unit is arbitrary [6]. Machines are subject to failures that are operation dependent [2,6], *i.e.* a machine can fail only while processing a part. The time to failure of machine M_i is assumed to be exponentially distributed with rate λ_i. The time to repair of machine M_i is exponentially distributed with rate μ_i.

Remark 1 *machines are numbered from* 1 *to* K. *In the rest of the paper we adopt the following convention: when* $i = K$, $i + 1 = 1$; *and when* $i = 1$, $i - 1 = K$.

In order to analyze this model, we approximate the behavior of the discrete asynchronous model by that of a continuous-flow model as described in [7]. Customers of the discrete model are now replaced by a continuous flow circulating from buffers to buffers. As long as it is able to do it, a machine transfers material from its upstream buffer to the downstream one at a constant rate $U = \frac{1}{T} = 1$. Failures and repairs have exactly the same characteristics as in the discrete model, meaning that they are both exponentially distributed with rates λ_i and μ_i for machine M_i. Now, as explained in [7], a choice must be made concerning the capacity of each buffer. Capacity of buffer $B_{i,i+1}$ in the continuous-flow model can indeed be chosen equal to the capacity of the associated buffer in the discrete model, or can alternatively be chosen to a capacity increased by one with regard to the one in the discrete model, in order to take into account the additional space available on the machine. This last choice is often made and has been proven to give better results [7]. In any case, we denote by $C_{i,i+1}$ the capacity of buffer $B_{i,i+1}$ in the continuous-flow model.

Machine M_i can be in one of the following states:

- working: it transfers material from its upstream buffer $B_{i-1,i}$ to its downstream buffer $B_{i,i+1}$ at a constant rate $U = 1$. As long as machine M_i is working it is subject to failures. The time to failure is exponentially distributed with rate λ_i;
- down: the machine cannot produce anymore and is under repair. The time to repair is exponentially distributed with rate μ_i;
- blocked: one machine M_j is down and all the buffers between M_i and M_j in the direction of the flow (namely $B_{i,i+1}, B_{i+1,i+2}, \ldots, B_{j-1,j}$) are full. When a machine is blocked it cannot fail;
- starved: one machine M_k is down and all the buffers between M_k and M_i in the direction of the flow (namely $B_{k,k+1}, B_{k+1,k+2}, \ldots, B_{i-1,i}$) are empty. When a machine is starved it cannot fail.

It is important to note that a machine cannot be in two of these states at a same time. For example, machine M_i cannot reach simultaneously the starved and the blocked state, this event corresponding to two independent simultaneous events, namely upstream buffer $B_{i-1,i}$ becoming empty (due to a failure of an upstream machine) and downstream buffer $B_{i,i+1}$ becoming full (due to a failure of a downstream machine). As a matter of fact, this event has a second order negligible probability. In addition, a machine can only fail when it is working.

We define for machine M_i the following parameters:

- e_i: efficiency of machine M_i in isolation: $e_i = \frac{\frac{1}{\lambda_i}}{\frac{1}{\lambda_i}+\frac{1}{\mu_i}} = \frac{\mu_i}{\mu_i+\lambda_i}$
- $I_i = \frac{\lambda_i}{\mu_i} = \frac{1}{e_i} - 1$
- pw_i: efficiency of machine M_i in the line; proportion of time machine M_i is working in line L;
- pf_i: proportion of time machine M_i is under repair in line L; pf_i can alternatively be considered as the probability of machine M_i to be under repair;
- ps_i: proportion of time machine M_i is starved in line L (starvation probability of machine M_i);
- pb_i: proportion of time machine M_i is blocked in line L (blocking probability of machine M_i);
- X_i: production rate of machine M_i in line L; as we assume $U = 1$:

$$X_i = Upw_i = pw_i \tag{1}$$

- $hm_{i,i+1}$: average amount of material in buffer $B_{i,i+1}$.

pw_i, pf_i, pb_i, ps_i, X_i and $hm_{i,i+1}$ are the performance parameters of the line, and thus the quantities that must be derived.

1.2 Basic equations

Some equations can be directly deduced from basic properties of the system. The flow conservation law implies that each machine must have the same production rate and the same efficiency:

$$X_i = X_j \quad \forall i, j \in \{1, \ldots, M\} \tag{2}$$
$$pw_i = pw_j \quad \forall i, j \in \{1, \ldots, M\} \tag{3}$$

As stated below, each machine is either working, under repair, starved, or blocked:

$$pw_i + pf_i + pb_i + ps_i = 1 \tag{4}$$

As long as machine M_i is working, it is subject to failure. The time to failure is exponential with rate λ_i. When it is under repair, the time to repair is exponential with rate μ_i. Therefore, the evolution of this system follows a two-state markovian process, whose equilibrium solution is given by [7]:

$$pw_i\lambda_i = pf_i\mu_i \tag{5}$$

From the previous relations, we can easily derive the following equation relating the efficiency of a machine in the line and its efficiency in isolation:

$$pw_i = e_i(1 - ps_i - pb_i) \tag{6}$$

Now, the sum of the amount of material in each machine must anytime be equal to N. This real-time constraint implies a constraint on the average amount of material:

$$\sum_{i=1}^{K} hm_{i,i+1} = N \tag{7}$$

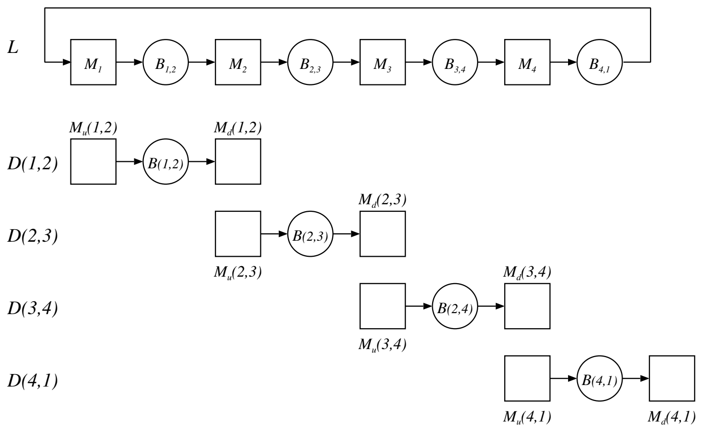

Fig. 2. Principle of the decomposition.

2 Decomposition

2.1 Principle

In this section we present the general decomposition principle of several approximation techniques that will be developed in the next section. The principle of the decomposition is to reduce the analysis of the initial line (L) to one of simpler systems that can be analyzed with exact techniques. We thus decompose the K-machine closed-loop line into a set of K *two-machine lines* $D(i,i+1)$ as illustrated in Figure 2. Each two-machine line $D(i,i+1)$ is a continuous-flow model consisting of an upstream machine $M_u(i,i+1)$, a downstream machine $M_d(i,i+1)$, and an intermediate buffer $B(i,i+1)$. The objective of the decomposition is to make the input/output flow through buffer $B(i,i+1)$ in $D(i,i+1)$ as close as possible from the input/output flow through buffer $B_{i,i+1}$ in the original line L. To do that, one must derive a set of equations as well as an algorithm, in order to estimate the parameters of the two-machine lines.

It is important to note that as we implicitly assume that there is an infinite supply of material in front of machine $M_u(i,i+1)$ and an infinite amount of storage available at the output of machine $M_d(i,i+1)$, machine $M_u(i,i+1)$ is never starved and machine $M_d(i,i+1)$ is never blocked. To fulfill the objective of the decomposition, namely to match the input/output flows through buffer $B(i,i+1)$ and buffer $B_{i,i+1}$, machine $M_u(i,i+1)$ must produce material in buffer $B(i,i+1)$ in a way as close as possible as machine M_i produces material

in buffer $B_{i,i+1}$. Now, machine M_i stops producing in three cases: 1) when it is under repair; 2) when it is starved, and 3) when it is blocked. This third case can be directly taken into account by $D(i,i+1)$, as a blocking of machine $M_u(i,i+1)$ corresponds to buffer $B(i,i+1)$ being full and machine $M_d(i,i+1)$ being under repair. On the other hand, machine $M_u(i,i+1)$ is never starved. The only way this machine can stop producing (excepted when it is blocked) is when it is under repair. We therefore understand why a failure of $M_u(i,i+1)$ must represent both a failure and a starvation of machine M_i. As a starvation of machine M_i is the consequence of a failure of an upstream machine in line L, we can thus state that machine $M_u(i,i+1)$ models the behavior of the upstream part of the line. Let us note that in the framework of a close-loop model, the "upstream" of buffer $B_{i,i+1}$ is made of all the machines of the line (from M_i, M_{i-1}, ..., to M_{i+1}) and all the other buffers (from $B_{i-1,i}$, $B_{i-2,i-1}$, ..., to $B_{i+1,i+2}$).

By a similar argument, a failure of machine $M_d(i,i+1)$ must represent both a failure and a blocking of machine M_{i+1}. Machine $M_d(i,i+1)$ will thus model the "downstream" part of the line, starting from buffer $B_{i,i+1}$, and including all the machines and all the buffers but $B_{i,i+1}$.

2.2 Characterization of the two-machine lines

In order to characterize a two-machine line $D(i,i+1)$, it is necessary to determine the capacity $C(i,i+1)$ of buffer $B(i,i+1)$, the production rate, as well as the failure/repair characteristics of machines $M_u(i,i+1)$ and $M_d(i,i+1)$. The capacity $C(i,i+1)$ will naturally be chosen equal to the capacity $C_{i,i+1}$ of the corresponding buffer $B_{i,i+1}$ in L. We assume that both machines of $D(i,i+1)$ have the same production rate as the machines in the original line, namely $U = 1$. Delays between failure and repair times are supposed to be described by exponential distributions with rates $\lambda_u(i,i+1)$ and $\mu_u(i,i+1)$ for machine $M_u(i,i+1)$, and $\lambda_d(i,i+1)$ and $\mu_d(i,i+1)$ for machine $M_d(i,i+1)$. Let us denote by $\mathcal{F}_u(i,i+1)$ the set of failure/repair parameters of machine $M_u(i,i+1)$ (namely $\lambda_u(i,i+1)$ and $\mu_u(i,i+1)$) and $\mathcal{F}_d(i,i+1)$ the set of failure/repair parameters of machine $M_d(i,i+1)$ ($\lambda_d(i,i+1)$ and $\mu_d(i,i+1)$).

We can now define the following parameters, similar to the one defined for the original line L, and where $e_u(i,i+1)$ and $e_d(i,i+1)$ are the efficiencies of machines $M_u(i,i+1)$ and $M_d(i,i+1)$ in isolation:

$$I_u(i,i+1) = \frac{\lambda_u(i,i+1)}{\mu_u(i,i+1)} = \frac{1}{e_u(i,i+1)} - 1 \tag{8}$$

$$I_d(i,i+1) = \frac{\lambda_d(i,i+1)}{\mu_d(i,i+1)} = \frac{1}{e_d(i,i+1)} - 1 \tag{9}$$

For each two-machine line there are 4 parameters to estimate. We therefore end up with a total of $4K$ unknowns. Let us first assume that all these unknowns have been estimated. Each two-machine line can thus be analyzed in isolation using the exact technique presented in [8]. The results of this analysis are the following performance parameters of the two-machine line in isolation:

- $p_w(i, i+1)$: efficiency of the two-machine line. The conservation flow law applied to the two-machine line actually implies that $p_w(M_u(i, i+1)) = p_w(M_d(i, i+1))$; We thus define $p_w(i, i+1) = p_w(M_u(i, i+1)) = p_w(M_d(i, i+1))$;
- $p_b(i, i+1)$: proportion of time the two-machine line is blocked. As the downstream machine of the two-machine line is never blocked $p_b(M_d(i, i+1)) = 0$; We thus define the blocking probability as the one of the upstream machine: $p_b(i, i+1) = p_b(M_u(i, i+1))$;
- $p_s(i, i+1)$: proportion of time the two-machine line is starved. As the upstream machine of the two-machine line is never starved $p_s(M_u(i, i+1)) = 0$; We thus define the starvation probability as the one of the downstream machine: $p_s(i, i+1) = p_s(M_d(i, i+1))$;
- $h_m(i, i+1)$: average amount of material in the two-machine line.

By applying relation (6) to machines $M_u(i, i+1)$ and $M_d(i, i+1)$ of $D(i, i+1)$ (and remembering that machine $M_u(i, i+1)$ is never starved and machine $M_d(i, i+1)$ is never blocked), we obtain the following relations:

$$p_w(i, i+1) = e_u(i, i+1)(1 - p_b(i, i+1)) \tag{10}$$
$$p_w(i, i+1) = e_d(i, i+1)(1 - p_s(i, i+1)) \tag{11}$$

Let us finally emphasize that, as soon as the performance parameters of every two-machine line have been obtained, each of them gives an estimate of the performance parameters of the original line L.

2.3 Equations

The system of equations is derived from the decomposition method on "open" production lines [4]. It consists in relating failure/repair parameters of machine $M_u(i+1, i+2)$ of $D(i+1, i+2)$ and those of machine $M_d(i-1, i)$ of $D(i-1, i)$ to the performance parameters of $D(i, i+1)$. This will be done by using several key relations. The first one states that the upstream machine of $D(i, i+1)$ and the downstream machine of $D(i-1, i)$ must have the same efficiency as machine M_i in line L:

$$pw_i = pw(i, i+1) = pw(i-1, i) \tag{12}$$

Let us note that from relation (3) we can deduce that all two-machine lines have the same efficiency. The second key relation prescribes that the downstream machine of $D(i-1, i)$ must have the same starvation probability as machine M_i in line L:

$$ps_i = ps(i-1, i) \tag{13}$$

The third key relation finally prescribes that the upstream machine of $D(i-1, i)$ must have the same blocking probability as machine M_i in line L:

$$pb_i = pb(i, i+1) \tag{14}$$

From equations (3), (6), (10), (11), (12), (13) and (14), we can easily derive the following relation (for more details about this derivation see [4]):

$$1 + I_d(i-1, i) + I_u(i, i+1) = \frac{1}{p_w(i, i+1)} + I_i = \frac{1}{p_w(i-1, i)} + I_i \tag{15}$$

The last key relation is directly inherited from the basic principle of the decomposition. Let us remember that a failure of machine $M_u(i+1, i+2)$ represents both a failure and a starvation of machine M_{i+1}. Let us denote by $\alpha_u(i+1, i+2)$ the proportion of failures of $M_u(i+1, i+2)$ that are not due to a failure of M_{i+1}. These failures are thus due to a starvation of M_{i+1}, this last being the consequence of either a failure or a starvation of machine M_i. These last two events are represented by a failure of machine $M_u(i, i+1)$. $\alpha_u(i+1, i+2)$ can thus be considered as the proportion of failures of $M_u(i+1, i+2)$ that are due to a failure of $M_u(i, i+1)$. Failure and repair times are exponentially distributed. From the memoryless property of the exponential distribution, we can thus obtain the mean repair time of machine $M_u(i+1, i+2)$ by the following relation:

$$\frac{1}{\mu_u(i+1, i+2)} = \frac{\alpha_u(i+1, i+2)}{\mu_u(i, i+1)} + \frac{(1 - \alpha_u(i+1, i+2))}{\mu_{i+1}} \tag{16}$$

$\alpha_u(i+1, i+2)$ must now be estimated. This parameter can be expressed as the ratio between the mean number of failures by unit of time of machine $M_u(i+1, i+2)$ that are due to a failure of machine $M_u(i, i+1)$, and the total number of failures by unit of time of machine $M_u(i+1, i+2)$. The detailed derivation of $\alpha_u(i+1, i+2)$ can be found in [4]. Here we only give the final result (17). It is important to note that the expression of $\alpha_u(i+1,i+2)$ only depends on the performance parameters of $D(i,i+1)$ and on $I_u(i+1,i+2)$.

$$\alpha_u(i+1,i+2) = \left[\frac{\mu_{i+1}}{\mu_u(i,i+1)}\left(\frac{I_u(i+1,i+2)pw(i,i+1)}{ps(i,i+1)} - 1\right) + 1\right]^{-1} \tag{17}$$

We finally obtain the system of equations characterizing the upstream machine:

$$\begin{aligned}
I_u(i+1,i+2) &= \frac{1}{pw(i,i+1)} + I_{i+1} - I_d(i,i+1) - 1 && (18)\\
\alpha_u(i+1,i+2) &= \left[\frac{\mu_{i+1}}{\mu_u(i,i+1)}\left(\frac{I_u(i+1,i+2)pw(i,i+1)}{ps(i,i+1)} - 1\right) + 1\right]^{-1} && (19)\\
\mu_u(i+1,i+2) &= \left[\frac{\alpha_u(i+1,i+2)}{\mu_u(i,i+1)} + \frac{(1-\alpha_u(i+1,i+2))}{\mu_{i+1}}\right]^{-1} && (20)\\
\lambda_u(i+1,i+2) &= I_u(i+1,i+2)\mu_u(i+1,i+2) && (21)
\end{aligned}$$

In a similar way we can derive the system of equations characterizing the downstream machine:

$$\begin{aligned}
I_d(i-1,i) &= \frac{1}{pw(i,i+1)} + I_i - I_u(i,i+1) - 1 && (22)\\
\alpha_d(i-1,i) &= \left[\frac{\mu_i}{\mu_d(i,i+1)}\left(\frac{I_d(i-1,i)pw(i,i+1)}{pb(i,i+1)} - 1\right) + 1\right]^{-1} && (23)\\
\mu_d(i-1,i) &= \left[\frac{\alpha_d(i,i+1)}{\mu_d(i,i+1)} + \frac{(1-\alpha_d(i,i+1))}{\mu_i}\right]^{-1} && (24)\\
\lambda_d(i-1,i) &= I_d(i-1,i)\mu_d(i-1,i) && (25)
\end{aligned}$$

Relations (18) to (25) are directly stemmed from the "open" line. They result in a set of $4K$ equations ($i = 1, \ldots, K$). Note that equations (18), (19), (22) and (23) are just given to simplify the writing of the system. In the context of a closed-loop line, it is shown in [9] that among these $4K$ equations, one is linearly dependent of the others, this dependency being the result of the loop. As a result the $4K$ equations form a set of $4K - 1$ independent equations. Thus we have to come up with one more independent equation. Actually, we have not derived yet an equation that involves the population constraint of the closed model, as expressed in (7). To take this constraint into account, we prescribe that the sum of the average amount of material in each two-machine line is equal to the total quantity of material available, *i.e.*:

$$\sum_{i=1}^{K} hm(i,i+1) = N \tag{26}$$

To sum up the results of this section, the system of equations is made of the set of relations (18) to (25), in which one relation has been removed and relation (26) has been added. Solving this system thus consists in choosing which one of the initial $4K$ equations has to be removed and how relation (26) is taken into account in the algorithm.

3 Solution methods

The basic principle of the solution methods comes from the "open" production lines and relies on a fixed point technique. As usual in numerical fixed point resolutions, the system is solved by starting with some initial values of the unknown parameters (parameters of the two-machine lines $D(i,i+1)$) and at each iteration of the algorithm by applying the equations of the system to the previously estimated parameters, in order to obtain better estimates. The process stops when parameters in between two successive iterations are close enough to each other. We then assume that convergence has been reached, even if no proof of it can be derived. The obtained parameters are supposed to be a good estimate of the unique solution of the system. Again, most of the time, no proof of the uniqueness of the solution can be driven.

3.1 Common principle

All the resolution techniques are based on a basic property of closed-loop production lines. This section presents this property. Let us first recall that the $4K$ equations of system (18) to (25) only forms a set of $4K - 1$ independent equations. Any equation of this system can thus be removed without changing the solution. If the equation to be removed is the one giving $I_u(i,i+1)$ (equation (18)) for a given value of i, the system turns out to be a system of $4K - 1$ equations with $4K - 1$ unknowns. Provided $I_u(i,i+1)$ is now given and considered as a fixed parameter, the system has a unique solution. Obviously, as equation (26) has not yet been taken into account, the solution of this system has no reason to fulfil the population constraint. This means that if an algorithm can solve this $4K-1$ independent equations system, the result obtained when the algorithm converges will be such that the sum of material in each two-machine line is not equal to N. We then define $Q(I_u(i,i+1))$ as the actual sum of material in each two-machine line when $I_u(i, i+1)$ has been fixed to a given value and when the previous system has been solved:

$$Q(I_u(i, i+1)) = \sum_{i=0}^{K} hm(i, i+1)$$

We can define similarly the quantity $Q(I_d(i, i+1))$ as the solution of the $4K$ equations system, in which one equation giving $I_d(i, i+1)$ (for a given value of i) has been removed and where $I_d(i, i+1)$ is set to a predetermined fixed value.

All methods are based on the following property:

For any i

- $Q(I_u(i, i+1))$ is a decreasing function of $I_u(i, i+1)$;
- $Q(I_d(i, i+1))$ is an increasing function of $I_d(i, i+1)$.

This property has never been demonstrated, but has always been verified experimentally (for more details see [9] and Section 3.3).

3.2 The initial method based on dichotomy

The method presented in this section is the original method developed by Frein, Commault and Dallery in [9]. The idea of this method is to apply a dichotomy to one of the parameters $I_u(i, i+1)$ or $I_d(i, i+1)$. They initially (and arbitrarily) chose to proceed on $I_u(1, 2)$.

The algorithm is made of an external loop and an internal loop. At each iteration of the external loop, the internal loop is called as a subfunction. The input of the internal loop is the value of the parameter $I_u(1, 2)$ that remains constant during all the iterations of a given internal loop. Now each iteration of the internal loop updates the two-machine lines parameters in two steps. A first step consists in updating the parameters of the upstream machines of all the two-machine lines, then a second step updates the parameters of the downstream machines in the reverse direction. At each iteration n of the internal loop, all two-machine lines are thus analyzed twice. When this procedure has converged, $Q(I_u(1, 2))$ is calculated, and becomes the output of the internal loop. The internal loop is referenced as Algorithm 1 and is given below.

The external loop uses the decreasing property of $Q(I_u(1, 2))$ with respect to $I_u(1, 2)$ and implements a dichotomy on an interval $[I_u(1, 2)_{inf}, I_u(1, 2)_{sup}]$

Algorithm 1 FCD: internal loop

Require: $I_u(1, 2)$
Initialize: $\lambda_u(i, i+1)$, $\lambda_d(i, i+1)$, $\mu_u(i, i+1)$ and $\mu_d(i, i+1)$ to the values of the corresponding machine in line L
Initialize: $\lambda_u(1, 2) = \mu_u(1, 2) * I_u(1, 2)$
repeat
  for $i = 1$ to K **do**
    evaluate the performance parameters of $D(i, i+1)$
    if $i \neq K$ **then**
      calculate $I_u(i+1, i+2)$ from (18)
    end if
    update $\lambda_u(i+1, i+2)$ and $\mu_u(i+1, i+2)$ from (19), (20) and (21)
  end for
  for $i = K$ to 1 **do**
    evaluate the performance parameters of $D(i, i+1)$
    calculate $I_d(i-1, i)$ from (22)
    update $\lambda_d(i-1, i)$ and $\mu_d(i-1, i)$ from (23), (24) and (25)
  end for
until convergence
$Q(I_u(1, 2)) = \sum_{i=1}^{K} hm(i, i+1)$

such that, at each iteration $Q(I_u(1, 2)_{inf}) \geq N$ and $Q(I_u(1, 2)_{sup}) \leq N$. The main algorithm (the external loop) is referenced as Algorithm 2 and will be denoted in the following "**FCD**".

This method is simple but has a few weaknesses. First, it is sometimes not feasible, even in real cases, to find values of $I_u(1, 2)_{inf}$ and $I_u(1, 2)_{sup}$ such that the corresponding $Q(I_u(1, 2))$ form an interval including N (algorithm 2, lines 2 to 14). Secondly, the use of two levels of iterations makes the method pretty inefficient. Finally, the choice of the machine used for making the dichotomy is very sensitive. We found many cases (corresponding to real situations) where a "bad" choice of the machine which parameter I_u (or I_d) is fixed in the inner-loop, makes the algorithm diverge. And there are unfortunately no heuristics to make a "good" choice of this machine. This dissymmetry in the method is thus basically problematic. In addition, we would like to extend the technique to much more general topologies.

3.3 Functions $Q = f(I)$

Most problems of the previous method actually come from the way the parameter driving the convergence ($I_u(i, i+1)$ or $I_d(i, i+1)$ for different values of i) is related to Q. In order to illustrate this problem, we give in Figure 3 on page 16 the curves $Q = f(I_u(i, i+1))_{i=1,\ldots,K}$ obtained for a 10-machine line

Algorithm 2 FCD: external loop

Require: N, ϵ
Initialize: $I_u(1,2)$ to a small value
repeat
  run the internal loop (algorithm 1)
  if $Q(I_u(1,2)) < N$ **then**
    choose a smaller value of $I_u(1,2)$
  end if
until $Q(I_u(1,2)) > N$
repeat
  $I_u(1,2)_{inf} = I_u(1,2)$ and $I_u(1,2)_{sup} = 2.I_u(1,2)$
  run the internal loop (algorithm 1) with $I_u(1,2) = I_u(1,2)_{sup}$
  if $Q(I_u(1,2)) > N$ **then**
    $I_u(1,2)_{inf} = I_u(1,2)_{sup}$ and $I_u(1,2)_{sup} = 2.I_u(1,2)_{sup}$
  end if
until $Q(I_u(1,2)) < N$
repeat
  run the internal loop with $I_u(1,2) = \frac{I_u(1,2)_{inf}+I_u(1,2)_{sup}}{2}$ (algorithm 1)
  if $Q < N$ **then**
    $I_u(1,2)_{sup} = I_u(1,2)$
  else
    $I_u(1,2)_{inf} = I_u(1,2)$
  end if
until $Q(I_u(1,2))$ is close enough to N (within ϵ)

Table 1
Parameters used in the $Q = f(I_u)$ study.

i	1	2	3	4	5	6	7	8	9	10
λ_i	0.02	0.07	0.02	0.1	0.5	0.005	0.07	0.03	0.02	0.001
μ_i	0.1	0.4	0.9	1.5	0.1	0.01	0.4	0.4	0.3	0.5
$C_{i,i+1}$	20	10	8	25	15	10	30	35	40	50

with parameters given in Table 1.

Remark 2 *the curves $Q = f(I_d(i,i+1))$ are very similar to the curves $Q = f(I_u(i,i+1))$ shown in Figure 3 on the following page except that they are increasing functions. This curves will not be given in this paper.*

We can see that each of these curves is made of a succession of "quasi-vertical" (high slope) parts and "quasi-flat" (low slope) parts. An algorithm realizing a search on the abscise parameter, will thus have to operate on the "flat" part in order to be efficient. If the search is carried out on the neighborhood of the "vertical" part, a very low variation of I_u will induce a strong variation of Q making the convergence procedure unstable. We now understand why the **FCD** algorithm doesn't converge in some cases, as it arbitrarily choses

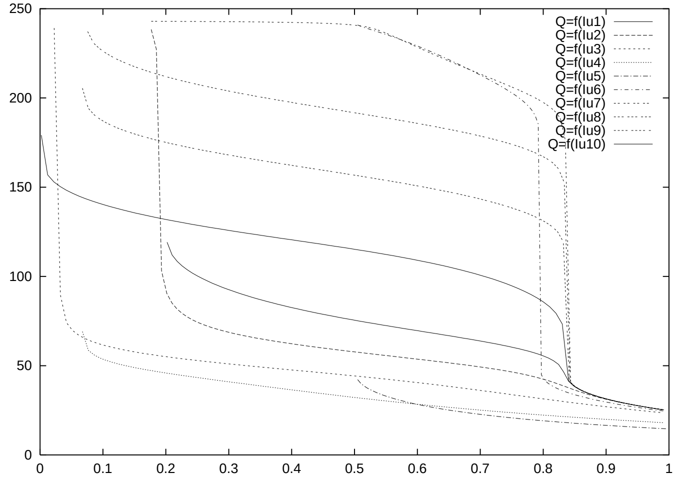

Fig. 3. $Q = f(I_u)$ for each machine of the line which parameters are given in Table 1 on the page before.

machine 1 to carry out the search. These divergent cases correspond to a "bad" choice of the machine, *i.e.* a vertical part in the neighborhood of N in the curve corresponding to machine 1. On the other hand, if the search is carried out on the "flat" part even a big variation of I_u will result in a small variation of Q and the algorithm will more likely converge. As a matter of fact, to obtained a robust and efficient algorithm, one has to choose the machine driving the convergence corresponding to the curve as "flat" as possible in the neighborhood of N. Unfortunately, there are no heuristics to a priori choose the "good" machine, this making the previous techniques hazardous to use.

3.4 A new simple loop method

The objective is both to eliminate the double-loop structure of the algorithm and the dissymmetry of the process related to the choice of a given machine, in order to give a more efficient and a more robust technique. To do that, we no longer fix a given parameter of the two-machine lines, but "drive" the convergence to the required final solution N, by imposing a constraint on a whole set of parameters. We transform the system of equation in order to add this constraint. Actually, we don't change the solution of the system, but the way the different equations of the system are taken into account in the resolution of the problem. Both systems (the initial one and the new one) will have exactly the same solution.

We modify relations (18) and (22) into:

$$I_u(i+1,i+2) = \beta_u \left(\frac{1}{pw(i,i+1)} + I_{i+1} - I_d(i,i+1) - 1 \right) \tag{27}$$

$$I_d(i-1,i) = \beta_d \left(\frac{1}{pw(i,i+1)} + I_i - I_u(i,i+1) - 1 \right) \tag{28}$$

The idea is to make the values of β_u (resp. β_d) dependent of Q/N (resp. N/Q) in the convergence process, in such a way that, when the algorithm converges, these two parameters are equal to 1 and we come back to the initial system. The objective of these corrective parameters is to increase the parameters I_u and to decrease the parameters I_d when $Q > N$, and to do the reverse when $Q < N$. This methods uses a parameter p in order to avoid oscillations in the convergence process. This parameter slows down convergence if at some point the algorithm tends to give a value of Q further from N than the one obtained in the previous iteration.

$$\beta_u = \left(\frac{Q}{N} \right)^{1/p} \tag{29}$$

$$\beta_d = \left(\frac{N}{Q} \right)^{1/p} \tag{30}$$

The algorithm is given in Algorithm 3 and referenced as "**DB**". In order to improve its efficiency, the algorithm recalculates the parameters β_u and β_d as soon as it needs them and from the most updated values of the performance parameters.

The underlying idea justifying the use of p is to "reduce" the "speed" of convergence in order to eliminate some oscillation problems. Indeed, we have observed on several examples some "under-damping" phenomenons introducing oscillations that drastically deteriorated the convergence of the algorithm. Therefore, it is necessary to detect early on when the algorithm "goes too fast". The easiest way is to detect when the value of Q in iteration $n+1$ has moved away from N with respect to its value in the previous iteration n. If this happens, we reduce the speed of the convergence by incrementing p. This idea has shown to be the more efficient one among all the policies that we tested and gets rid of oscillating problems in almost all tested examples.

To update the value of Q in lines 12 and 21 of Algorithm 3, we just "correct" its value by subtracting the value of the average amount of a two-machine line before analyzing it in isolation and adding the updated value of it after

Algorithm 3 DB

1: **Initialize:** $\lambda_u(i,i+1)$, $\lambda_d(i,i+1)$, $\mu_u(i,i+1)$ et $\mu_d(i,i+1)$ to the values of the corresponding machine in line L
2: **Initialize:** β_u and β_d to 1.0
3: **Initialize:** p to 1
4: **Initialize:** Q_{old} to 0
5: **Initialize:** $iter$ to 0
6: **repeat**
7:   **for** $i = 1$ to K **do**
8:     evaluate the performance parameters of $D(i,i+1)$
9:     calculate $I_u(i+1,i+2)$ from (27)
10:     update $\lambda_u(i+1,i+2)$ and $\mu_u(i+1,i+2)$ from (19), (20) and (21)
11:     **if** $iter > 0$ **then**
12:       update Q
13:       $\beta_u = \left(\frac{Q}{N}\right)^{1/p}$
14:     **end if**
15:   **end for**
16:   **for** $i = K$ to 1 **do**
17:     evaluate the performance parameters of $D(i,i+1)$
18:     calculate $I_d(i-1,i)$ from (28)
19:     update $\lambda_d(i-1,i)$ and $\mu_d(i-1,i)$ from (23), (24) and (25).
20:     **if** $iter > 0$ **then**
21:       update Q
22:       $\beta_d = \left(\frac{N}{Q}\right)^{1/p}$
23:     **end if**
24:   **end for**
25:   $Q = \sum_{i=1}^{K} hm(i,i+1)$
26:   $iter = iter + 1$
27:   **if** $|Q - N| > |Q_{old} - N|$ **then**
28:     $p = p + 1$
29:   **end if**
30:   $Q_{old} = Q$
31:   $\beta_u = \left(\frac{Q}{N}\right)^{1/p}$
32:   $\beta_d = \left(\frac{N}{Q}\right)^{1/p}$
33: **until** convergence of all the parameters

the analysis in isolation. In order to eliminate numerical error propagation problems we fully recalculate Q at the end of each iteration (step 25).

4 Numerical results

In this section we will compare the new technique (**DB** presented in Section 3.4) to the previous one (**FCD** presented in Section 3.2) for aspects of both speed and robustness using several specific and random examples.

4.1 Generated examples

It is important to emphasize that, in this paper, we only compare the "qualitative" performance of the algorithms, such as the speed of convergence and the robustness. Indeed, as stated before, the system of equations that is solved by the different techniques is exactly the same as the one originally proposed in [9], and has already been discussed in terms of "quantitative" performance, such as the the average amount of material in each buffer and the production rate of the different machines.

The speed of convergence is measured by calculating the number of analyses in isolation of all the two-machine lines, necessary to reach the convergence for the different techniques. As mentioned in the previous sections, the algorithms may not converge to the solution (due to numerical problems). We, thus, limited the number of analyses in isolation to a predetermined value that we fixed to 20.000. This limit clearly appears on several graphics given below. An algorithm that has reached this limit stops and a flag indicates that, for the tested configuration, the algorithm failed and thus gives fake results (that do not correspond to the solution of the system). Let us note that in some cases, by increasing this limit, one can make the algorithm converge, whereas there are cases for which the algorithm will fail whatever the limit is. We measure the robustness as the percentage of time a given algorithm fails to reach the convergence. This parameter has been obtained from random tests and the results are presented below.

4.1.1 Symmetrical examples

We first ran totally symmetrical examples. Our experience has shown that this is the most favorable case for the two algorithms. All the machines have exactly the same failure/repair parameters: $\lambda_i = 0.01$ and $\mu_i = 0.1$, for any $i = 1, \ldots, K$. All the buffers have exactly the same capacity: we tested two different capacities, namely $C_{i,i+1} = 2$ and $C_{i,i+1} = 10$, for any $i = 1, \ldots, K$. We also ran the different algorithms on two different sizes of the line: $K = 3$ and $K = 10$. The results are summarized in Figure 4 on the following page. Each of the 4 graphs in this figure corresponds to a couple (C, K), where C denotes in a short way the capacity of each buffer of the line. The X-axis is the

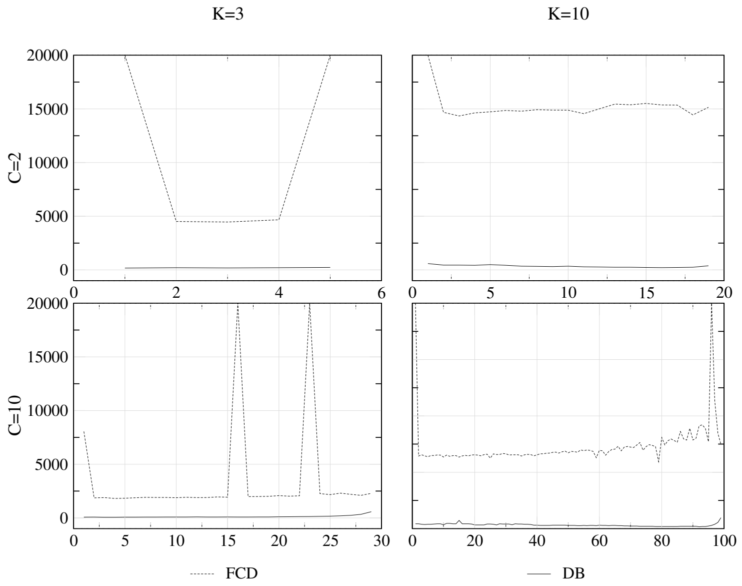

Fig. 4. Example 1: Symmetrical examples.

number N of pallets in the line. This number varies from 1 up to $C_{tot}-1$, where C_{tot} is the total capacity of the line, $C_{tot} = C \times K$. The Y-axis is the number of analyses in isolation of all the two-machine lines, for the three techniques.

First, we must note the poor performance of the original **FCD** algorithm, as this last needs from around 5.000 to 10.000 analyses in isolation to reach the convergence in most cases, and sometimes does not converge at all. In turns, the new **DB** algorithm always converges quickly to the solution, usually 10 to 100 times (or more) faster than the original **FCD** technique.

4.1.2 Random examples

In order to measure the robustness and the general behavior of the techniques, we generated random examples. All the random parameters are generated from a uniform distribution on a given interval. For any machine M_i of the line we thus drew the parameters μ_i and I_i (from which we obtained λ_i) and for any buffer $B_{i,i+1}$ we drew a capacity $C_{i,i+1}$. We used a uniform distribution in the following intervals:

- $\mu_i \in [0.01, 1.0]$

Table 2
Example 2.

i	1	2	3	4	5
λ_i	0.018	0.010	0.055	1.6e-3	0.017
μ_i	0.220	0.250	0.590	0.087	0.350
I_i	0.081	0.041	0.092	0.018	0.048
$C_{i,i+1}$	2	12	16	23	24

Table 3
Example 3.

i	1	2	3	4	5	6	7	8	9	10
λ_i	0.016	0.074	0.086	0.081	0.020	1.3e-3	0.063	8.3e-3	0.089	0.093
μ_i	0.230	0.840	0.920	0.810	0.420	0.026	0.880	0.750	0.990	1.000
I_i	0.070	0.088	0.094	0.100	0.047	0.048	0.071	0.011	0.090	0.094
$C_{i,i+1}$	24	27	21	12	29	12	25	29	13	9

- $I_i \in [0.01, 0.1]$
- $C_{i,i+1} \in \{1, \ldots, 30\}$

We ran 2.000 random examples. A first set of 1.000 random examples corresponds to lines with $K = 5$ machines and a second set of 1.000 examples corresponds to lines with $K = 10$ machines. Each of these examples is tested for all population from 1 up to $C_{tot} - 1$, where C_{tot} is the total capacity of the given configuration. In addition, we present two kinds of results:

- the results of one particular example chosen among the 1.000 randomly generated examples, to show a typical behavior of the different techniques;
- the mean results of the 1.000 random examples, to show the robustness of the different techniques.

Specific example. The parameters of the line (chosen among the random examples) consisting of $K = 5$ machine are given in Table 2 and referred to as Example 2, those corresponding to a line with $K = 10$ machines are given in Table 3 and referred to as Example 3. The total capacity of Example 2 is 77 and the total capacity of Example 3 is 201.

The numerical results for Example 2 and 3 are given in Figures 5 and 6. We note that the difference between the previous technique (**FCD**) and the new one (**DB**) is much bigger than in the case of a totally symmetrical example (Example 1). Moreover, the initial technique fails to converge towards the solution in many cases (represented on the graphics by a maximum at 20.000), whereas the new technique always converges in less than 500 analyses in isolation. More specifically, for $K = 5$ (Figure 5 on the next page), the previous technique has a very irregular behavior, and for $K = 10$ (Figure 6 on the following page) it fails to converge on a very large range of N (actually any

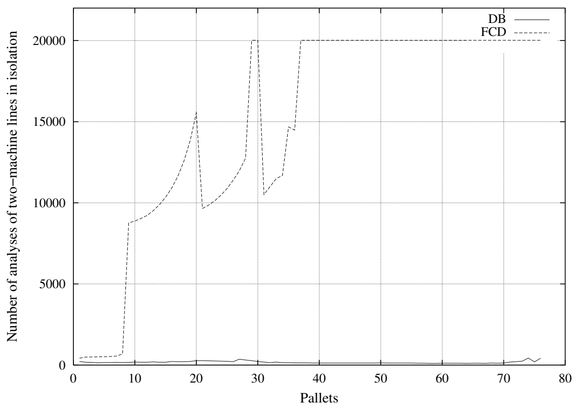

Fig. 5. "Speed" of convergence for Example 2 (Table 2 on the page before).

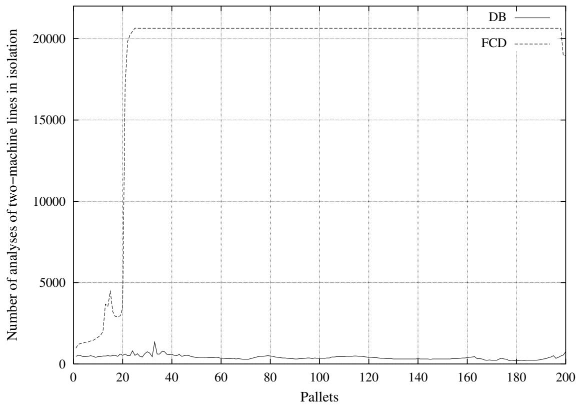

Fig. 6. "Speed" of convergence for Example 3 (Table 3 on the preceding page).

value of N greater than 25).

Statistic results. The results summarizing the 2.000 random examples (1.000 examples for $K = 5$ and 1.000 examples for $K = 10$) are presented in Figures 7 and 8. The Y-axis of these curves represents now the average number of analyses in isolation of all the random examples. As the total capacity of the different examples differs, the X-axis represent the relative value of N, meaning the ratio $\frac{N}{C_{tot}}$ expressed in percentage of the total capacity C_{tot} of each example. In addition, some percentage numbers are indicated on the

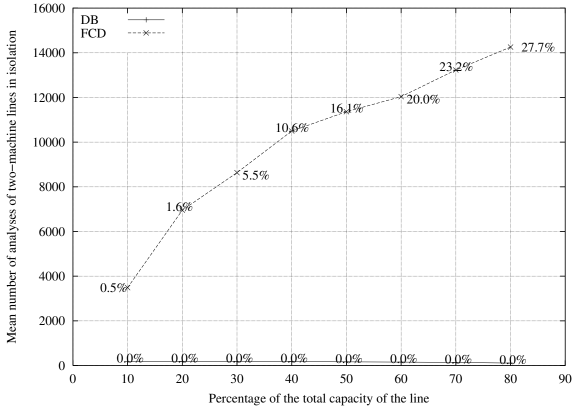

Fig. 7. Statistics for variable parameters lines with $K = 5$.

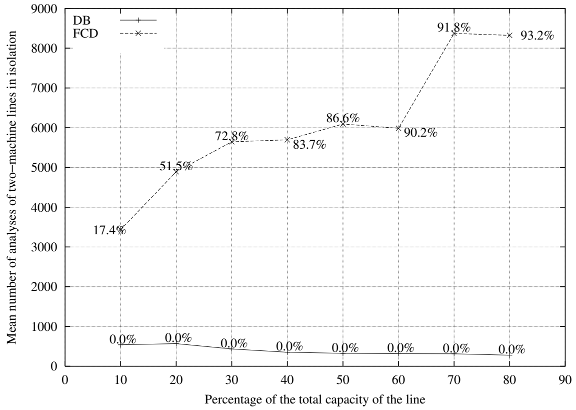

Fig. 8. Statistics for variable parameters lines with $K = 10$.

curves and correspond to the percentage of tests (among the 1.000 ones and corresponding to a given relative value of N) that did not converge, *i.e.* that reached the limit of 20.000 analyses in isolation. It is important to note that the 20.000 limit values are not taken into account in the average values and are thus not "part of" the curves.

The conclusions are very clear, both on efficiency and robustness of the new technique:

- efficiency: if we only concentrate on the "central" region (between 30% to 70% of the total capacity), the new technique (**DB**) is between 30 and

more than 100 times faster than the original one (**FCD**). Moreover we only consider here the cases that have converged.
- robustness: on all the examples the new technique has not failed a single time, whereas the previous one failed in 10% of the cases for $K = 5$, when the number of pallets is half the capacity of the line, and failed in more than 80% of the cases for $K = 10$.

Conclusion

In this paper we have presented a new analytical method for estimating the performance parameters of a closed-loop production line with unreliable machines and finite buffers. The technique solves the same system of equations as the original method proposed in [9], but in a much more efficient and robust way. The following conclusions can be drawn:

- the previous technique [9] fail to converge to the solution of the system of equations in many situations, including practical cases;
- the converging problems come from the double-loop structure of the previous algorithms and from numerical instabilities; these problems are explained and detailed on several examples;
- the new technique avoid these problems by proposing a single-loop structured algorithm and has proven to converge quickly in almost all situations, including examples with a very high variability in parameters;
- the new technique is thus much more efficient, even on examples for which the previous techniques converge, and much more robust;
- in addition, by avoiding the dissymmetrization and the double-loop structure of the previous algorithms, it can be more easily extended to the resolution of manufacturing systems with much more general topology (including routing and Fork/Join mechanisms); these extension are currently under experiment.

References

[1] Akyildiz I.F. On the exact and approximative throughput analysis of closed queueing networks with blocking. *IEEE*, 14:62–70, 1988.

[2] J. A. Buzacott and L. E. Hanifin. Models of automatic transfer lines with inventory banks - A review and comparison. *AIIE Transactions*, 10(2):197–207, 1978.

[3] J. A. Buzacott and J. G. Shanthikumar. *Stochastic Models of Manufacturing Systems*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[4] Y. Dallery, R. David, and X.-L. Xie. An efficient algorithm for analysis of transferlines with unreliable machines and finite buffers. *IIE Transactions*, 20:280–283, 1988.

[5] Y. Dallery, R. David, and X.-L. Xie. Approximate analysis of transfer lines with unreliable machines and finite buffers. *IEEE Transactions on Automatic Control*, 34(9):943–953, 1989.

[6] Y. Dallery and S. B. Gershwin. Manufacturing flow line systems: A revies of models and analytical results. *Queueing Systems*, 12:3–94, 1992.

[7] R. David, X. Xie, and Y. Dallery. Properties of continuous models of transfer lines with unreliable machines and finite buffers. *IMA J. of Mathematics in Business and Industry*, 6:281–308, 1990.

[8] D. Dubois and J.-P. Forestier. Productivité et en-cours moyen d'un ensemble de deux machines séparées par une zone de stockage. *RAIRO Automatique*, 16(2):105–132, 1982.

[9] Y. Frein, C. Commault, and Y. Dallery. Modeling and analysis of closed-loop production lines with unreliable machines and finite buffers. *IIE Transactions*, 28:545–554, 1996.

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

[11] N. Maggio. An analytical method for evaluating the performance of closed loop production lines with unreliable machines and finite buffer. Master's thesis, Politecnico di Milano, 2000.

[12] N. Maggio, A. Matta, S.B. Gershwin, and T. Tolio. Analysis of loop networks by decomposition. In *Third Aegean International Conference on Design an Analysis of Manufacturing Systems*. University of Aegean, May 2001.

[13] R.O. Onvural and H.G. Perros. Some equivalences between closed queuing networks with blocking. *Performance Evaluation*, 9:111–118, 1989.

[14] R. Suri and Diehl G.W. A variable buffer size model and its use in analyzing closed queuing networks with blocking. *Management Sciences*, 32:206–224, 1986.

[15] L. Zhuang, J.A. Buzacott, and X.-G. Liu. Approximation mean-value analysis of cyclic queuing networks with production blocking. *QUESTA*, 16:139–165, 1994.