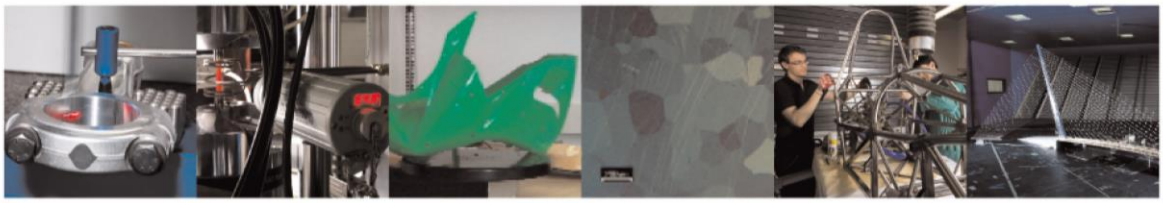


A contribution to operations management-related issues and models for home care structures

Sahin, Evren; Matta, Andrea

This is an Accepted Manuscript of an article published by Taylor & Francis in INTERNATIONAL JOURNAL OF LOGISTICS on 13 Oct 2014, available online:
http://www.tandfonline.com/10.1080/13675567.2014.946560

DIPARTIMENTO DI MECCANICA ■ POLITECNICO DI MILANO
via G. La Masa, 1 ■ 20156 Milano ■ EMAIL (PEC): pecmecc@cert.polimi.it
http://www.mecc.polimi.it
Rev. 0

Integrated simulation–optimisation of pull control systems

Giulia Pedrielli[a], Arianna Alfieri[b] and Andrea Matta[c*]

[a]*Centre For Maritime Studies, National University of Singapore, Singapore, Singapore;* [b]*Dipartimento di Ingegneria Gestionale e Produzione, Politecnico di Torino, Turin, Italy;* [c]*Department of Industrial Engineering and Management, School of Mechanical Engineering, Shanghai Jiao Tong University, Shanghai, China*

(*Received 4 June 2014; accepted 30 November 2014*)

1. Introduction

Pull control policies are considered to be more effective and robust than push strategies mainly due to their ability to quickly react to unpredictable changes in customer demand and to their ease of implementation (Deleersnyder et al. 1989; Hopp and Spearman 2007; Song 2013; Spearman and Zazanis 1992). Indeed, pull policies typically control the *work in process* (WIP), which is more observable and less sensitive than the throughput rate, i.e. the measure traditionally considered by push policies (Hopp and Spearman 2007).

Kanban Control System (KCS) and Base Stock Control System (BSCS) are among the most implemented control policies. They are fully characterised by a single parameter (*base stock level* and the number of *kanban tokens*, respectively) and are widely used in the industrial practice (Bonvik, Couch, and Gershwin 1997). Specifically, BSCS uses the base stock level as target inventory position at a specific stage. This objective requires generation of a production order every time a customer demand is received, immediately transferring the demand information to all stages (*global information* on demand). As a result, the system is highly reactive. However, the maximum number of jobs in the system at any time is not limited. On the contrary, KCS uses *local information* to control the maximum number of jobs circulating in the system. Indeed, the number of kanban tokens at each stage (the policy parameter) corresponds to the maximum number of jobs that can simultaneously stay between each pair of servers in the system. As a result, a job can be prevented from entering a stage if no kanban token is available and a stage token becomes available only when a job leaves. The described stage-by-stage backward propagation causes a low reactivity of the system, that is particularly exacerbated when the number of kanban tokens is small (Liberopoulos and Dallery 2000).

The Extended Kanban Control System (EKCS) is a generalisation of KCS and BSCS policies. Using two parameters per stage, EKCS tries to provide reactivity while limiting the number of jobs in the system (Dallery and Liberopoulos 2000). To this end, both the number of kanban tokens and the base stock level represent control parameters for the policy.

In this paper, we deal with the problem to design control policies, i.e. given the control strategy to apply (e.g. KCS, BSCS or EKCS), we need to optimally identify the value of the related parameters (Rouwenhorst et al. 2000). This problem is interesting and non-trivial for two main reasons: (1) once a policy has been chosen, the problem of selecting the values of the parameter(s) is usually combinatorial. Furthermore, the underlying system is in general stochastic both in terms of service times as well as customer demand; (2) different control policies are generally hard to compare. Each policy minimises a specific cost function while honouring some performance target levels (e.g. average lateness, customer miss rate, system throughput, system WIP and waiting time), which typically vary based on the specific policy. Since these performance are in trade-off, a good policy should be able to balance conflicting objectives. However, the evaluation of the best achievable performance is generally not a trivial task.

*Corresponding author. Email: matta@sjtu.edu.cn

In order to overcome these issues, this work proposes *integrated* simulation–optimisation models that simultaneously determine the control parameters of the policy under analysis (KCS, BSCS and EKCS policies are considered in this work), while considering the system dynamics. The novelty of the approach, with respect to the current literature, resides in the ability to perform the two tasks (policy parameter optimisation and performance evaluation) using the same model, which is distribution-free (i.e. independent from the specific distribution used to model uncertainty) and flexible enough to be modified for providing the estimate of the maximum achievable performance thus, handling the second issue previously outlined.

The remainder of the paper is structured as follows: Section 2 provides details on the main novelties of this work with respect to the existing literature. Section 3 introduces the technique adopted to approximate the mathematical models for the integrated simulation–optimisation of KCS, BSCS and EKCS policies, whose functioning is illustrated in Section 4. The integrated models are the subject of Section 5 together with the benchmark models. The experimentation to investigate the performance of the approach is presented in Section 6. Finally, Section 7 draws the conclusions of the paper.

2. Literature review and contribution

In the literature, both analytical and heuristic approaches have been proposed for solving the problem of setting the policy parameters in the case of stochastic pull controlled systems (Daniel and Rajendran 2006; Shang and Song 2007; Shang, Song, and Zipkin 2009).

Mathematical programming models have been developed within the scope of optimisation to represent the system dynamics, i.e. how 'objects' and 'information' flow through the system, subject to physical and logical constraints (Aglan and Durmusoglu 2014; Helber, Schimmelpfeng, and Stolletz 2011; Helber et al. 2011; Shang and Song 2007; Shang, Song, and Zipkin 2009). Cottenceau, Hardouin, and Ouerghi (2008) propose the use of (max, $+$) algebra to optimise KCS, with the objective of increasing the service level and decreasing the amount of WIP.

Queueing models have been proposed for estimating the performance of all the most popular and well-known pull policies, such as KCS (Di Mascolo, Frein, and Dallery 1996; Huang and Kusiak 1996; Matta, Dallery, and Di Mascolo 2005; Park and Lee 2013) and BSCS (Duri, Frein, and Di Mascolo 2000; Liberopoulos and Dallery 2000). However, to the knowledge of the authors, no analytical method has been developed to assess the performance of the EKCS.

Stochastic control theory and dynamic programming have also been proposed to design optimal policies for stochastic manufacturing systems (Ohno 2011; Yan, Yin, and Lou 1994).

Recently, simulation–optimisation has received noticeable attention, especially for KCS and BSCS (Cochran and Kaylani 2008; Fu 2002; Gaury, Kleijnen, and Pierreval 2001; Kleijnen 2008; Köchel and Nieländer 2002; Koulouriotis, Xanthopoulos, and Tourassis 2010; Yang, Fu, and Yang 2007). In fact, discrete event simulation has been widely adopted for performance evaluation of pull control systems (Hao and Shen 2008). In this field of application, simulation–optimisation is a family of iterative procedures, where a search module is required to generate the candidate control parameter values while a simulation module assesses the related performance (Bonvik, Couch, and Gershwin 1997; Fu 2002; Gaury, Pierreval, and Kleijnen 2000; Geraghty and Heavey 2005; Karaesmen and Dallery 2000; Oladipupo et al. 2014; Smew, Young, and Geraghty 2013; Wang and Shi 2013). This technique requires far fewer assumptions with respect to queueing theory. However, a large number of iterations is required to reach a good solution, and it is characterised by poor asymptotic properties.

This paper stems from the approach presented in Alfieri and Matta (2012a) and Pedrielli (2013) presenting new results in the optimisation of the parameters for the control policies KCS, BSCS and EKCS. Specifically, we propose, for the first time, integrated simulation–optimisation models to solve the policy design problem for this subclass of control strategies. For each policy, a single mathematical model is developed, having as decision variables policy parameters as well as event times. The former are part of the objective function (usually a cost or a system performance), whereas the event times are used to develop the constraints reproducing the system dynamics and to include the realisations of service times and arrival times (i.e. demand rates). The result is a mixed integer programming (MIP) model and the returned solution is the optimal parameter configuration for the policy of interest (with respect to the generated set of arrivals and service times).

To reduce the required computational effort, the MIP model is linearised using the time buffer approximation technique proposed in Alfieri and Matta (2012a) and Pedrielli (2013) (refer to Section 3 for additional details). According to this approx-imation, all the synchronisation mechanisms between jobs, typical of pull control policies, are translated into synchronisation mechanisms between events. This translation is such that complex mathematical models with integer decision variables are easily commuted into linear mathematical models, reducing the computational effort to solve the policy design problem.

The present work differs from the previous research under two main aspects: (1) It proposes and implements for the first time integrated simulation optimisation models for the solution of the design problem for KCS, BSCS and EKCS. Mathematical programming models have already been proposed to simulate (but not optimising) pull policies. Alfieri and Matta (2012b) propose mathematical models to estimate the performance of the main pull control policies applied to single-product serial manufacturing systems. However, the developed models are used only in the scope of performance evaluation,

whereas, only a few guidelines are provided on their extension for optimisation purposes. The mathematical programming model for the optimisation of extended kanban systems (EKCS) is formulated in Alfieri and Matta (2012a), but no experimental analysis is performed. (2) *Benchmark models* are devised to ease the policy comparison, i.e. to evaluate the quality of the policy when several performance measures are considered. These models consider the system dynamics within the constraints and use as objective function the minimisation (maximisation) of the performance of interest. The solution of the benchmark model is an ideal policy that cannot be implemented. Nevertheless, it provides a bound (lower or upper) on the achievable performance, increasing the effectiveness of the evaluation of the proposed control strategy. Benchmark models replace intensive simulation analysis which is usually required to estimate the best system performance.

3. Integrated simulation–optimisation through time buffer approximation

Recently, an alternative framework for simulation–optimisation has been proposed (Alfieri and Matta 2012a, 2013; Pedrielli 2013). This approach stems from the work of Chan and Schruben (Chan and Schruben 2003, 2008a, 2008b; Schruben 2000), who propose mathematical models having as objective the minimisation of the sum of event times. The constraints model the system dynamics (e.g. demand arrival, jobs process sequence and assembly operations) and consider the system static properties (e.g. limited buffer capacities (Chan and Schruben 2003, 2008b), constant number of customers (Alfieri, Matta, and Pedrielli in press; Chan and Schruben 2008a)) as well as policies governing the system (e.g. finite buffer blocking mechanisms, priorities, maximum sojourn times). The solution of these models is the simulated sample path of the stochastic discrete event system under analysis. In other words, feeding a traditional simulator with the same random numbers, we would obtain exactly the same system trajectory.

These mathematical models are linear for a large category of systems, but this characteristic is lost, most of the times, if mathematical programming models are used for optimisation, which makes mathematical programming less attractive in the scope of simulation–optimisation. Moreover, once the MIP is solved, it provides a sample path optimal solution (i.e. for a specific generation of random numbers), and it is difficult to perform sensitivity studies on the solution under the MIP framework. To overcome these difficulties, Alfieri and Matta (2012a) propose an approximation method based on the *time buffer* concept. The time buffer allows a complex MIP problem to be transformed into a simple LP problem that can also be used to derive a good approximation of the optimal solution to the original MIP.

The basic idea behind the time buffer is that it approximates the 'space-based' blocking mechanism through a time-based relationship (Alfieri and Matta 2012a). Let us consider two jobs a and b to be served in the sequence $a \rightarrow b$ at a stage j of a serial production system. Assume that job b has to be processed at stage h, whereas job a undergoes its process at stage j and $j > h$. Job b is blocked by job a if its starting time at stage h is greater than or equal to the finishing time of job a at stage j.

The advantage of the time-based blocking interpretation is that it can be easily relaxed by adjusting the interval between event times. Indeed, when this interval is positive, the starting time of customer b on server h can be anticipated as follows:

$$x_{bh} \geq y_{aj} - s,$$

where x_{bh} is the starting time of customer b at stage h, y_{aj} is the finishing time of customer a at stage j and s is the *time buffer*. Job b at stage h can start s time units before the service of job a is completed at stage j. However, job a can block job b if $s = 0$. When applying the presented concept, it is essential to consider that the time buffer-based blocking represents an approximation of the space-based blocking (Alfieri and Matta 2013; Pedrielli 2013). Nevertheless, time buffer approximate models still represent a valid approach for simulation–optimisation because only LP models need to be solved. Moreover, the relationships between approximate LP models and exact MIP models can be exploited to find an approximate solution to the exact problem through a formal mapping of the time buffer to the original domain of the problem (i.e. the space of the policy parameters).

The time buffer approximation has been tested with the buffer allocation problem (BAP) for open queueing networks and to size the number of customers in a closed queueing network (Alfieri and Matta 2012a; Alfieri, Matta, and Pedrielli in press; Schwarz and Stolletz 2013; Stolletz and Weiss 2013). For the BAP case, the characterisation of the approximate model asymptotic properties has also been developed (Pedrielli 2013).

4. Analysed control policies

As already stated in the previous sections, in this work, we focus on three main control policies: KCS, BSCS and EKCS that will be described in the following sections. The integrated models for the simultaneous simulation–optimisation of the proposed policies will be the subject in Section 5.

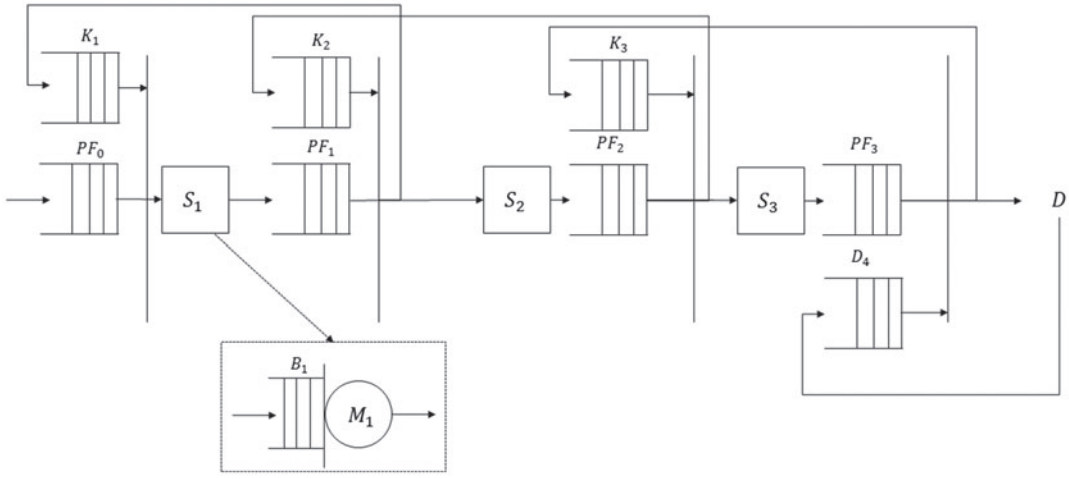


Figure 1. Three-stage Kanban Control System.

4.1 *Kanban Control System*

A multi-stage queueing network managed by a kanban policy is represented in Figure 1, where the number of stages is $J = 3$. The additional stage $j = J + 1 = 4$ refers to the *demand stage*, i.e. the point at which jobs are released to satisfy demand (stage D in Figure 1).

Each stage S_j is associated with a fixed number of kanban tokens k_j and it is composed of a server M_j ($j = 1, \ldots, J$) and the related input buffer (B_j), which is assumed to have an infinite capacity. A *synchronisation station* between each pair of adjacent stages ($j, j + 1$) graphically represents the coordination mechanism. The synchronisation station has two incoming queues: (1) K_{j+1} containing the kanban tokens authorising jobs to be released to stage $j + 1$ and (2) PF_j containing the finished parts produced by S_j. The jobs in queue PF_j are attached a kanban token related to stage j (whose related queue is K_j); if kanban tokens are available in queue K_{j+1}, they can be attached to jobs in PF_j that are then released to stage $j + 1$, whereas the attached kanban token is sent back to the related queue K_j (synchronisation mechanism). As a result, if K_{j+1} (PF_j) is empty, jobs (kanban tokens) are blocked and wait in the queue until a kanban (job) arrives.

At the last stage, $J + 1$, the logical queue D_{J+1} (D_4 in Figure 1 (Liberopoulos and Dallery 2000)) contains the external demands. The optimisation problem can be formulated as the minimisation of the number of kanban tokens required to meet a target performance level. In particular, we will consider as performance the average lateness, which is computed as:

$$\alpha = \frac{1}{n} \sum_{i=1}^{n} \left(z_{i,J+1} - d_i \right), \tag{1}$$

where $z_{i,J+1}$ represents the time when the ith job is released from the system and d_i the time when the demand for the ith job is realised.

We will refer to this optimisation problem as the Optimal Kanban Allocation Problem (OKAP).

4.2 *Base stock control system*

Figure 2 represents a three-stage line controlled by a Base Stock Control System, where the number of stages is $J = 3$. The demand stage is modelled as in the KCS case.

As the KCS, the BSCS can be fully characterised by a single control parameter for each stage j, i.e. the base stock level, b_j. The stock quantity b_j affects the way the production system is 'initialized'. More specifically, it decreases the system lateness. However, it does not have any impact on the system efficiency (i.e. the system throughput (Di Mascolo, Frein, and Dallery 1996)).

Each stage S_j is associated with the target inventory level, i.e. the base stock b_j. It is composed of a server M_j ($j = 1,\ldots, J$) and the related input buffer (B_j), which is assumed to have an infinite capacity. A *synchronisation station* between each pair of adjacent stages ($j, j + 1$) graphically represents the coordination mechanism. The synchronisation station j has two incoming queues: (1) D_{j+1}, containing the demand tokens coming from the $J + 1$th stage authorising jobs to be released to stage $j + 1$ and (2) PF_j, containing the finished parts produced by S_j. At the simulated time 0, PF_j contains b_j parts, and

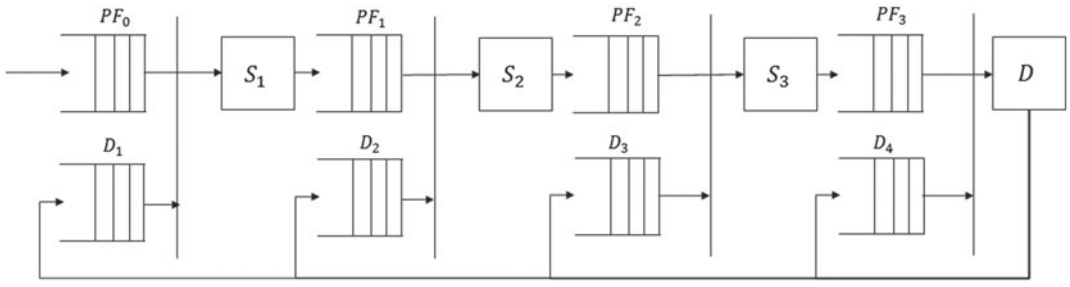


Figure 2. Three-stage Base Stock Control System.

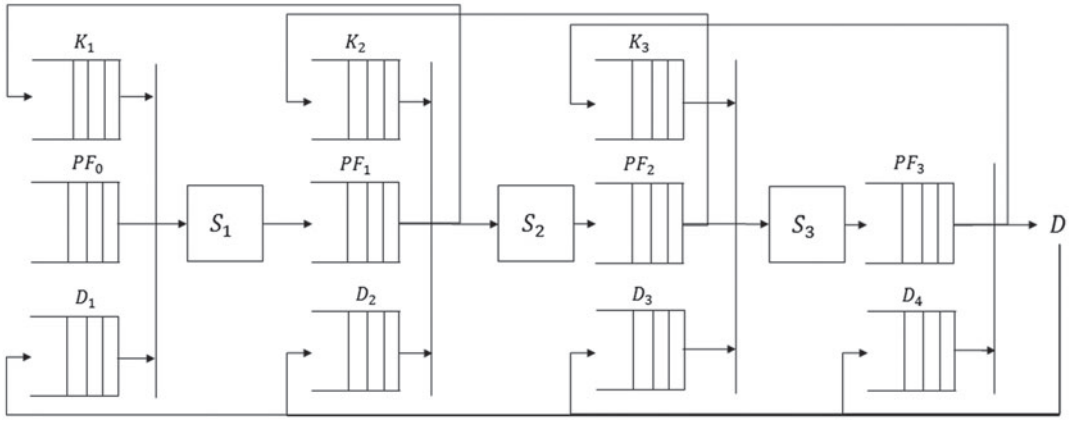


Figure 3. Three-stage Extended Kanban Control System.

queues D_j are all empty. As soon as a demand order arrives, it is transferred to all queues D_j (global information). The jobs in queue PF_j are released to the next stage $(j+1)$ if D_j is not empty. As a result, if D_{j+1} (PF_j) is empty, jobs (demand) are blocked and wait in the queue until a demand order (job) arrives.

In the last stage, $J+1$, D_{J+1} (D_4 in Figure 1 (Liberopoulos and Dallery 2000)) contains the external demands.

The considered optimisation problem is the minimisation of the total base stock costs. As in the kanban case, the policy has to guarantee the average lateness to be less than a predefined threshold as modelled in Equation (1). We will refer to this optimisation problem as the Optimal Base Stock Allocation Problem (OBSAP).

4.3 *Extended Kanban Control System*

A multi-stage queueing network managed by the extended kanban policy is represented in Figure 3, where the number of stages is $J = 3$. The demand stage is modelled as in the previous cases.

The synchronisation station is such that a job is released only when both the kanban (Section 4.1) and base stock (Section 4.2) conditions for authorising jobs are contemporaneously satisfied. In other words, the EKCS dynamics is the result of the union of the KCS and BSCS dynamics.

5. Integrated models for simulation–optimisation

In this section, the models for the solution of the OKAP, OBSAP and EKCS optimisation will be detailed. The main notation and terminology preliminary to the models presentation will be provided in Section 5.1. Section 5.2 will present the integrated simulation–optimisation models whose structural properties will be investigated in Section 5.3. The benchmark models will be the subject in Section 5.4.

The mathematical programming models are deterministic, since they are solved for a specific set of realisations of the random variables involved in the process. If the models are solved multiple times, we will always obtain different solutions. Moreover, there are no closed-form analytical expressions to evaluate the system performance. The set of equations presented in this section describe the rules characterising the system dynamics.

It is noteworthy that, in the scope of optimisation, time buffer models can currently approximate (refer to Alfieri and Matta (2012a) for more details): (1) systems with identical jobs; (2) split or merge systems and (3) assembly and disassembly systems.

The main criticality of the proposed models is the fact that the mathematical formulations consider a fixed job sequence. To have a fixed sequence is, however, irrelevant in case jobs are identical. Relaxing the assumption of identical jobs leads to the need to separate job position from job identity if fixed sequence is relaxed too. This means to generate variables of the type x_{ikj} , i.e. the event time of job i located in the kth position in the job sequence and starting its process on stage j. The main condition for the presented models to be valid is that the ordering of the jobs does not change or if it does, jobs are identical.

According to the presented approach, the control parameters (number of kanban tokens and base-stock levels) are identified according to a specific sample path, i.e. through a single run optimisation procedure. The underlying idea is that, as long as the number of jobs forming the sample path is large enough, the solution obtained should be close to the 'true' optimal solution (i.e. resulting from the infinite sample path problem) as long as the demand process and the processing times are stationary and remain consistent with the distribution used to generate the sample path for the optimisation.

5.1 *Main notation and terminology*

We consider a production system composed of J single-server stages through which n identical jobs flow. The jobs flow to satisfy customer orders that arrive at the system. We assume that no scheduling decision is made and that the First In First Out rule is applied. Servers are assumed to be perfectly reliable (or, alternatively, repair times are considered negligible).

For simplicity, each job i ($i = 1,...,n$) is supposed to be available at simulated time 0, and, after its release to the system, it is processed by the first machine ($j = 1$). Although we assume that each customer order requires exactly one job, we need both terms *customer* and *job* to separate the stochastic process of the demand arriving at the system (i.e. customers),

which is an *exogenous* process, from the items processed by the machines (i.e. jobs), which represent an *endogenous* process.

Variables $\{z_{ij}\}$ represent the earliest release times of job i ($i = 1, \ldots, n$) to stage j ($j = 1, \ldots, J + 1$), where stage $J + 1$ represents the *demand stage*. Variables $\{x_{ij}\}$ are the starting event times of jobs i ($i = 1, \ldots, n$) at the machine j ($j = 1, \ldots, J$). Variables $\{y_{ij}\}$ represent the time job i ($i = 1, \ldots, n$) leaves the jth machine ($j = 1, \ldots, J$) to enter stage $j + 1$ or to leave the system ($j = J$). While starting times $\{x_{ij}\}$ and finishing times $\{y_{ij}\}$ can be found in any mathematical programming model of multi-stage queuing networks, regardless of the control policy, the release times $\{z_{ij}\}$ are typical of controlled system. In these systems, the release of a part to a stage is disjointed from its processing, i.e. a part can be completed and ready to enter a stage, but its release can be delayed (usually to prevent the system congestion).

Customer arrival times $\{d_i\}$ and processing times at each stage $\{t_{ij}\}$ are problem parameters that are generated from the stochastic process $\{D_i\}, i = 1, \ldots, n$, and from the sequence of stochastic processes $\{T_i\}_j, i = 1, \ldots, n, j = 1, \ldots, J$, respectively. Since no assumptions are required on the adopted stochastic processes, problem parameter values can be sampled from any distribution; alternatively, if available, they can be collected from the real system. Notice that, since the problem parameters are indeed sampled from stochastic processes, the models presented in the following correspond to a single simulation run.

5.2 *Integrated simulation–optimisation models*

In this section, we present the MIP and approximate LP models for solving the OKAP, OBSAP and the combined problem for the EKCS.

5.2.1 *KCS integrated model*

The MIP model solving the OKAP problem can be formulated as follows:

$$\min \sum_{j=1}^{J} \left(\sum_{k=1}^{k_j^{up}} 1_{kj} \cdot k \right) \tag{2}$$

s.t.

$$x_{i+1,j} - y_{ij} \geq 0 \quad i = 1, \ldots, n-1, j = 1, \ldots, J \tag{3}$$

$$x_{i,j} - z_{ij} \geq 0 \quad \forall i, j = 1, \ldots, J \tag{4}$$

$$z_{i,j+1} - y_{ij} \geq 0 \quad \forall i, j = 1, \ldots, J \tag{5}$$

$$z_{i,J+1} \geq d_i \quad \forall i \tag{6}$$

$$y_{i,j} - x_{ij} \geq t_{ij} \qquad \forall i, j = 1, \ldots, J \tag{7}$$

$$\frac{1}{n} \sum_{i=1}^{n} \left(z_{i,J+1} - d_i \right) \leq \alpha^* \tag{8}$$

$$z_{i+k,j} - z_{i,j+1} \geq \left(1_{kj} - 1 \right) \cdot M \qquad j = 1, \ldots, J \quad k = 1, \ldots, k_j^{up}, i = 1, \ldots, n-k, \tag{9}$$

$$\sum_{h=1}^{k_j^{up}} 1_{hj} = 1 \qquad j = 1, \ldots, J \tag{10}$$

The binary decision variables $\left\{1_{kj}\right\}$ $k = 1, \ldots, k_j^{up}$, $j = 1, \ldots, J$ take value 1 if a number of kanban tokens equal to k is assigned to stage j and 0 otherwise; k_j^{up} represents the maximum number of allowed kanban tokens at stage j. The objective is to minimise the total number of kanban tokens allocated to the line (2).

A machine cannot process two different parts at the same time (constraints (3)) and a part cannot start the production in a stage if no kanban tokens are available (constraints (4)). Constraints (5) force the authorisation to move job i to stage $j+1$ to occur after its process completion at stage j. Equations (7) states that the process duration of job i at stage j ($y_{ij} - x_{ij}$) cannot be smaller than the processing time t_{ij}. Constraint (8) imposes that the average lateness of the system is not larger than the maximum allowed lateness α^*. Equations (9) represent the (local) backward information flow due to the kanban control mechanism. If k is the number of kanban tokens assigned to stage j, job $i+k$ will be allowed to enter stage j only when part i is released to stage $j+1$. We use the big-M method to model this synchronisation. The trigger of this backward flow is the customer demand, as stated by constraints (6). Constraints (10) state that only one kanban level can be chosen for each stage. Constraints (3)–(6) can be referred to as *production constraints*, and they are common to every pull system independently of the specific policy adopted. Finally, all the continuous decision variables x, y and z must be non-negative.

As stated in the previous sections, this model is hard to solve. Nevertheless, we can linearise it by applying the time buffer approximation (Section 3). In order to do so, the binary variables $\left\{1_{kj}\right\} k = 1, \ldots, k_j^{up}$ $j = 1, \ldots, J$ are replaced by the continuous variables $\left\{s_{kj}^K\right\} k = 1, \ldots, k_j^{up}$ $j = 1, \ldots, J$ representing the time buffer allocated to stage j between parts having a lag smaller than k. The MIP objective is approximated by the minimisation of the total amount of time buffer allocated to the system. The resulting LP can be formulated as follows (OKAP model):

$$\min \sum_{j=1}^{J} \left(\sum_{k=1}^{k_j^{up}} s_{kj}^K \right) \tag{11}$$

s.t.

$$(3) - (8)$$

$$z_{i+k,j} - z_{i,j+1} \geq -s_{kj}^K \quad j = 1, \ldots, J \quad k = 1, \ldots, k_j^{up} \quad i = 1, \ldots, n-k \tag{12}$$

Equations (12) replace (9). The earliest time part $i+k$ will be allowed to enter stage j is the time part i is released to stage $j+1$ minus the time buffer s_{kj}^K. With respect to the general definition in Section 3, in the KCS case, the time buffer, approximating the kanban level at each stage, can be allocated to modify the interval between the two events *release of job from stage j* and *release of job from stage j + 1*.

5.2.2 *BSCS integrated model*

Also for the BSCS case, we derive both MIP and LP models to solve the OBSAP. The MIP optimisation model results as follows:

$$\min \sum_{j=1}^{J} \left(\sum_{b=1}^{b_j^{up}} 1_{bj} \cdot b \right) \tag{13}$$

s.t.

$$(3) - (6); (8)$$

$$y_{i,j} - x_{ij} \geq t_{ij} - \left(1 - \sum_{b=1}^{i-1} 1_{bj}\right) \cdot M \quad i = 1, \ldots, b_j^{up} \;\; j = 1, \ldots, J \tag{14}$$

$$y_{i,j} - x_{ij} \geq t_{ij} \qquad i = b_1^{up} + 1, \ldots, n \qquad j = 1, \ldots, J-1 \tag{15}$$

$$z_{i+k,j} \geq d_i - \left(1 - \sum_{b=1}^{k} 1_{bj}\right) \cdot M \qquad j = 1, \ldots, J \qquad k = 1, \ldots, b_j^{up} \;\; i = 1, \ldots, n-k \tag{16}$$

$$\sum_{b=1}^{b_j^{up}} 1_{bj} = 1 \qquad j = 1, \ldots, J-1 \tag{17}$$

In place of $\left\{1_{kj}\right\}$, $k = 1, \ldots, k_j^{up}$ $j = 1, \ldots, J$, we define the binary decision variables $\left\{1_{bj}\right\}$ $b = 1, \ldots, b_j^{up}$ $j = 1, \ldots, J$ to model the echelon base stock level at each stage. In particular, the variable $1_{bj} = 1$ if an echelon base stock equal to b is selected for stage (echelon) j. The echelon base stock at stage j can vary between 1 and b_j^{up}.

The objective function (13) is the minimisation of the sum of the base stock levels ($1_{bj} \cdot b$), selected for each stage.

Constraints (14) and (15) model the synchronisation mechanism characterising BSCS (Section 4.2). In particular, constraints (14) represent the initialisation. Job i, where $i \in [1, \ldots, b_1^{up}]$ (note that b_1^{up} represents the base stock of the whole line), will be processed at stage j only if the echelon base stock b_j allocated to that stage is such that $b_j < i$; otherwise, it means that job i is in stock in one of the buffers $PF_l, l \geq j$ and, therefore, it does not have to undergo any process. Constraints (15) force each part $i \in [b_1^{up} + 1, \ldots, n]$ to leave the stage after being processed by the server. Constraints (16) model the base stock (global) information flow: as soon as demand for a part occurs, this information is transferred to all the stages, allowing the release of a job from the stage (if all other constraints are satisfied). Finally, the selection of a unique echelon base stock level for each stage is guaranteed by constraints (17). Also in this case, we proceed with the time-buffer approximation obtaining the following approximated model:

$$\min \sum_{j=1}^{J} \left(\sum_{b=1}^{b_j^{up}} s_{bj}^e \right) \tag{18}$$

s.t.

(3) − (6); (8); (15)

$$y_{i,j} - x_{ij} \geq t_{ij} - s_{ij}^e \;\; j = 1, \ldots, J-1 \;\; i = 1, \ldots, b_j^{up} \tag{19}$$

$$z_{i+k,j} \geq d_i - s_{kj}^e \qquad j = 1, \ldots, J-1 \qquad k = 1, \ldots, b_j^{up} \;\; i = 1, \ldots, n-k \tag{20}$$

Continuous time buffers (decision variables) $\left\{s_{ij}^e\right\}$ $i = 1, \ldots, b_j^{up}$ $j = 1, \ldots, J$ are introduced to approximate the already-defined decision variables 1_{bj} . With respect to the general definition in Section 3, in the BSCS case, the time buffer that approximates the base stock level at each stage can be allocated to modify the interval between the two events *release of job from stage j* and *customer demand arrival*.

5.2.3 *EKCS integrated model*

The formulation proposed herein is different from that provided in (Alfieri and Matta 2012a) because the decision variables represent the echelon base stock and not the installation base stock. The optimisation problems are equivalent because we are optimising a serial production line with a single job type and single-location echelon (Axster and Rosling 1993). Nevertheless, the notation changes as the echelon-based view considerably simplifies the model structure. The MIP optimisation model is:

$$\min \sum_{j=1}^{J} \left(\left(\sum_{b=1}^{b_j^{up}} 1_{bj} \cdot b \right) + \left(\sum_{k=1}^{k_j^{up}} 1_{kj} \cdot k \right) \right)$$

s.t.

$$(3) - (6);\ (8);\ (9);\ (14) - (16)$$

$$\sum_{b=1}^{s_j^{up}} 1_{bj} = 1 \quad j = 1, \ldots, J$$

$$\sum_{k=1}^{k_j^{up}} 1_{kj} = 1 \quad j = 1, \ldots, J \tag{21}$$

All the constraints are taken from the kanban and base stock models, and the objective function (21) minimises the sum of kanban tokens and the sum of base stocks assigned to the system.

We proceed with the linearisation, obtaining the following model:

$$\min \sum_{j=1}^{J} \left(\left(\sum_{h=1}^{s_j^{up}} s_{hj}^{e} \right) + \left(\sum_{k=1}^{K_j^{up}} s_{kj}^{K} \right) \right)$$

s.t.

$$(3) - (6);\ (8);\ (12);\ (19) - (20) \tag{22}$$

As in the case of the exact integrated model, the optimisation model combines the approximate optimisation model for the kanban time buffer and the base stock time buffer. The objective function (22) minimises the sum of both the kanban and base stock time buffers.

Mathematical models can be developed to evaluate the performance of the optimised system (Alfieri and Matta 2013; Chan and Schruben 2008b; Matta 2008). In particular, given the solution in terms of number of kanban tokens or time buffer, we can feed a mathematical model with the same random numbers used in the optimisation step. The result of this mathematical model is the system trajectory under the generated random numbers, i.e. it is equal to the result of a single replication of the traditional discrete event simulation. It should be noted that the sample path generated solving the simulation model is such that the obtained event times represent upper bound to those obtained in the real system when the policy is applied. As a result, the simulation model is not useful in the sole scope of optimising the system of interest. According to the proposed approach, the simulation is run only for the purpose of computing the exact performance, but it is not actually relevant to optimise the policy control parameters.

Also, a single goal programming approach might be adopted and solved minimising a weighted sum of the time buffer and the event times, in place of the proposed models. Nonetheless, the single goal programming model poses a key issue: setting the appropriate weight for the time buffers and the finishing times. Indeed, an a priori structural rule for setting the weights is not available. This poses a problem in stating the optimality properties of the obtained solution as we are not able to characterise it in terms of its relationship to the minimum time buffer and maximum system throughput. On the contrary, the solution of the approximate optimisation model represents a structural lower bound to the throughput of the system, i.e. without the need of running the approximate simulation, we can guarantee that the system will attain at least the target throughput under the generated configuration.

The obtained time buffer configuration is then adopted to derive the approximate solution in terms of the integer value of the control parameters at each stage of the line. This configuration will be referred to as $\tilde{k}_j,\ j = 1, \ldots, J$, $\tilde{b}_j,\ j = 1, \ldots, J$ and $\left(\tilde{k}_j, \tilde{b}_j\right),\ j = 1, \ldots, J$ for the KCS, BSCS and EKCS, respectively. The *approximate integer solution* can be easily computed by exploiting the structural characteristics of the time buffer matrix. In particular, we use the fact that when the time buffer between two jobs having lag k (i.e. jobs i and $i + k$) equals 0, the control parameter must be at least k. The approximate integer solution can then be computed as:

$$\tilde{k}_j = \text{argmax}_{k=1,\ldots,k_j^{up}} k : s_{kj} > 0,\ j = 1, \ldots, J \tag{23}$$

$$\tilde{b}_j = \text{argmax}_{k=1,\ldots,b_j^{up}} k : s_{bj} > 0,\ j = 1, \ldots, J \tag{24}$$

The simulation models of the analysed systems were presented in Alfieri and Matta (2012b) and are not reported in this paper.

5.3 *Structural properties*

In this section, we characterise the presented models in terms of the properties of the solutions. The complete proofs can be found in Appendix 1.

5.3.1 *KCS policy*

The time buffer resulting from the KCS integrated simulation–optimisation model can be characterised both with respect to the lag k (Property 1) and to the target performance α^* (Property 2).

Property 1 (Kanban Time Buffer) The optimal sequence $\left\{s_{kj}^{*K}\right\}$ obtained solving the OKAP model is non-increasing in k.

Property 1 is fundamental for the computation of the approximate integer kanban level corresponding to the optimal time buffer matrix. Moreover, the value of each element of the time buffer is strongly related to the correlation between jobs being processed in the system with respect to the target performance. In particular, the lower the value of the time buffer, the lower the effect of the correlation of the two jobs on the system performance. In addition, the way the time buffer values decrease as the lag k increases can also suggest guidelines for the selection of the kanban levels.

Property 2 The optimal sequence $\left\{s_{kj}^{*K}\right\}$ obtained by solving the OKAP model is non-increasing in the target performance α^*.

5.3.2 *BSCS policy*

Also for BSCS, as in the case of KCS, it is possible to characterise the optimal time buffer with respect to the lag k (Property 3) and to the target performance α^* (Property 4).

Property 3 (Base Stock Time Buffer) The optimal sequence $\left\{s_{bj}^{*e}\right\}$ obtained by solving the OBSAP model is non-increasing in k.

Property 4 The optimal sequence $\left\{s_{bj}^{*e}\right\}$ obtained by solving the OBSAP model is non-increasing in the target performance α^*.

5.3.3 *EKCS policy*

It is possible to derive the following property relating the optimal time buffer to the kanban and to the base stock.

Property 5 The optimal kanban time buffer at the last stage s_{kJ}^{K} is such that:

$$s_{kJ}^{K} > s_{kJ}^{e} \quad \forall k$$

This result is particularly relevant as it confirms the analytical result in Dallery and Liberopoulos (2000), i.e. the number of kanban tokens is always greater than or equal to the base stock levels (Property 2 in the referenced paper, page 374).

5.4 *Benchmark models*

Most of the times, when designing and analysing a policy, the underlying objective is to maximise the system performance, e.g. service level, system throughput, system WIP or average waiting time. These performance are in trade-off and a good policy is the one that balances conflicting objectives. The problem of quantifying the quality of a policy is not trivial. Ideally, we might want to know the best achievable performance level for each of the aforementioned criteria and compare it with the one obtained when the system is governed by the proposed policy. However, except for simple cases, closed-form expressions are not available to compute the system performance.

The mathematical programming models for integrated simulation–optimisation proposed in this paper can be further exploited to easily gather an estimate of the aforementioned bounds on the system performance, with the resulting advantage of having a benchmark against which we can compare the designed policy.

The models to obtain the system bounds are referred to as *benchmark models* and can be, in general, formulated as follows:

$$\min \vartheta \tag{25}$$

$$\text{s.t.}$$

$$\mathbf{D}(\mathbf{xyz})' \geq (\mathbf{d}, \mathbf{t})' \tag{26}$$

In this model, ϑ represents the performance measure: the system throughput, a function of the waiting time or the WIP in the system; in case costs are defined at each stage of the system, function (25) represents a cost.

Constraints (26) describe the system dynamics (e.g. precedence between jobs, process sequences, finite buffer capacities, etc.), but they do not include the policy characterisation, neither performance constraints as the performance is in the objective function. Specifically, $\mathbf{D}$ represents the matrix of coefficients multiplying the transpose vector $(\mathbf{xyz})'$ containing starting, departure and release times. The right hand side $(\mathbf{d}, \mathbf{t})$ refers to the demand process $\mathbf{d}$ and the processing times $\mathbf{t}$.

As an example, consider a pull system in which we want to minimise the total cost related to the time spent by a job at a certain stage of the system. In this case, the objective function (25) will be detailed as:

$$\sum_{j=1}^{J} \sum_{i=1}^{n} c_j \left(z_{i,j+1} - z_{i,j} \right), \tag{27}$$

where $\mathbf{c} = c_1, \ldots, c_J$ is a known vector of costs, one for each stage j. The general constraints (26) will be instead detailed as:

$$x_{i+1,j} - y_{ij} \geq 0 \; i = 1, \ldots, n-1, \; j = 1, \ldots, J \tag{28}$$

$$x_{i,j} - z_{ij} \geq 0 \qquad \forall i, \; j = 1, \ldots, J \tag{29}$$

$$z_{i,j+1} - y_{ij} \geq 0 \qquad \forall i, \; j = 1, \ldots, J \tag{30}$$

$$z_{i,J+1} \geq d_i \qquad \forall i \tag{31}$$

$$y_{i,j} - x_{ij} \geq t_{ij} \qquad \forall i, \; j = 1, \ldots, J \tag{32}$$

The solution of this model is the minimum cost achievable by the system under an ideal policy consisting in deciding the release time z_{ij} for each job at each stage in the production line. Clearly, this policy cannot be implemented in reality since processing times and, hence, starting, completion and release times, are not decision variables. Nonetheless, the benchmark model gives us a key information: which is the best system cost and the optimal system sample path to reach that cost. This fact is of fundamental relevance not only to support the evaluation of a policy, but also when designing the control policy.

6. Numerical results

Numerical experiments on randomly generated instances have been performed to validate the time buffer approximation when applied to the optimisation of the parameters of the analysed pull control policies. In particular, we have the following objectives: (i) evaluate the quality of the approximate integer solution against a commercial optimiser (providing the optimal solution) and (ii) exploit the benchmark models to investigate their potentiality for supporting the design and optimisation of the control policy.

The focus of the experiments is on the analysis of the approximate integer solution computed from the optimal time buffer matrix. In particular, we first provide an evaluation of the quality of the solutions obtained from the three policies. Then, we compare the solutions in terms of kanban tokens and base stock levels when optimising these parameters separately (i.e. when adopting KCS and BSCS, respectively) and when the EKCS is used (i.e. the parameters are optimised together).

The approximate integer solution has been compared with the output obtained from the commercial optimiser OptQuest© embedded within the simulation software Arena©. In this experiment, OptQuest© is treated as the benchmark, i.e. it is assumed that it provides the true optimal solution. Hence, the absolute difference between the approximate integer solution and OptQuest© solution is referred to as a measure of *absolute error*.

We will use the following notation:

- KCS: The approximate solution is referred to as $\tilde{\mathbf{k}} = (k_1, k_2, \ldots, k_J)$, and the related objective function value is $\kappa = \sum_{j=1}^{J} k_j$. The same measures obtained from OptQuest© are referred to as $\mathbf{k}^* = \left(k_1^*, k_2^*, \ldots, k_J^*\right)$ and $\kappa^* = \sum_{j=1}^{J} k_j^*$, respectively.

- BSCS: The approximate solution is referred to as $\tilde{\mathbf{b}} = (b_1, b_2, \ldots, b_J)$, and the related objective function value is $\mathcal{B} = \sum_{j=1}^{J} b_j$. The same measures obtained from OptQuest© are referred to as $\mathbf{b}^* = \left(b_1^*, b_2^*, \ldots, b_J^*\right)$ and $\mathcal{B}^* = \sum_{j=1}^{J} b_j^*$, respectively.
- EKCS: The approximate solution is referred to as $\tilde{\mathbf{e}} = ((k_1, s_1), (k_2, s_2), \ldots, (k_J, s_J))$, and the related objective function value is $\mathcal{E} = \sum_{j=1}^{J} \left(k_j + s_j\right)$. The same measures obtained from OptQuest© are referred to as $\mathbf{e}^* = \left(\left(k_1^*, b_1^*\right), \left(k_2^*, b_2^*\right), \ldots, \left(b_J^*, b_J^*\right)\right)$ and $\mathcal{E}^* = \sum_{j=1}^{J} \left(k_j^* + b_j^*\right)$, respectively.

Simulation–optimisation has been executed for several independent random instances. The same generated instances were also used to feed standard simulation models developed in Arena©. The results obtained from the standard simulation were the same as those obtained from the mathematical programming models for exact simulation, thus, validating their correctness.

All the experiments were carried out on a Core i5 2.53 Ghz CPU and the CPLEX 12.1 *Interior Point* algorithm was used to solve the LP models. We observed that the computational time required by the approximate model to solve the instances of the different problems goes from the order of the seconds when small instances are considered (2 to 3 stages) to minutes in case of larger instances (up to 7 stages). The solution time required by OptQuest© is orders of magnitude larger (20 to 30 minutes for small instances up to 3 hours for 8 stages). The exact MIP model reveals the worst performance requiring more than 10 hours to solve a single instance with 7 stages.

The computational time is influenced by both the number of stages (J) characterising the problem instance and the number of parts n. For example, the MIP model is not able to solve instances with 8 stages if more than 1000 jobs need to be considered, thus making the use of MIP models impractical.

6.1 *Experimental settings*

The system we tested is characterised by $J = 3$ stages, $j = J + 1 = 4$ being the demand stage. Pilot experiments were performed up to $J = 7$ stages, showing the robustness of the approach to problems of larger size, as the results will show. The multiple replication optimisation was performed over sample paths of $n = 5000$ jobs and a maximum expected lateness value $\alpha^* = 0.1$. The processing times were generated from an exponential distribution with mean 1.

Four experimental conditions were designed, characterised by exponential customer inter-arrival times with different means $1/\lambda = (2.0, 1.665, 1.33, 1.25)$.

For each experimental condition, 10 independent replications were run using $\min \mathbf{s}^K$, $\min \mathbf{s}^e$ and $\min\left(\mathbf{s}^K + \mathbf{s}^e\right)$ as objective functions when optimising the number of kanban tokens for KCS, the base stock level for BSCS and the kanban and base stock levels for EKCS, respectively. Each experimental condition replicate was tested on KCS, BSCS and EKCS by adopting common random numbers.

In addition to the approximate integer solution value, we computed the absolute error at each stage as:

$$e_j = |v_j^* - \tilde{v}_j|, \quad j = 1, \ldots, J$$

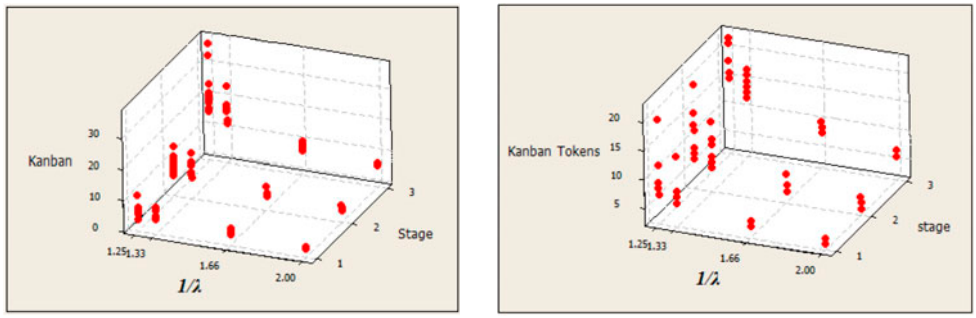
where v_j^* represents the optimal value for the policy parameters obtained from OptQuest©, whereas $\tilde{v}_j$ represents the approximate value obtained from a single replication of the approximate optimisation model. The error was computed for each component of the approximate integer solution (e.g. $\tilde{k}_j \; \forall \; j$ in the case of KCS) and for the related value of the objective function. This results in 30 error measures for the solution components (60 in the case of EKCS) and 10 error measures for the objective function for each value of $1/\lambda$, for a total of 480 error measures for the solution components and 120 for the objective function.

6.2 *Approximate model performance*

We start analysing the approximate solution resulting from the time buffer model under the different tested scenarios.

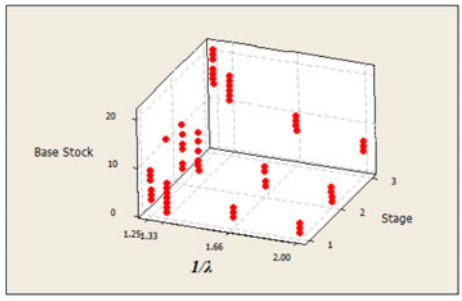
Figure 4 shows how kanban level and base stock level increase towards the final stage of the system, and as the inter-arrival time decreases. This behaviour happens in the KCS case (Figure 4(a)), in the BSCS case (Figure 4(c)) and in the EKCS case (Figure 4(b) and 4(d)) and it is consistent with the literature (Dallery and Liberopoulos 2000; Di Mascolo, Frein, and Dallery 1996; Duri, Frein, and Di Mascolo 2000; Liberopoulos and Dallery 2000). In fact, as the arrival rate increases, in order to guarantee the target performance, more kanban tokens are needed in the KCS; more base stock is needed to increase the system reactivity in the BSCS; in the EKCS, more base stock is needed to increase the system reactivity and more kanban tokens are required to increase the flow of parts through the production system.

As already discussed in Section 1, the presence of more parameters is one of the characteristics that makes the EKCS more effective, especially in the case of high demand rate or high demand variability. This phenomenon can be seen in

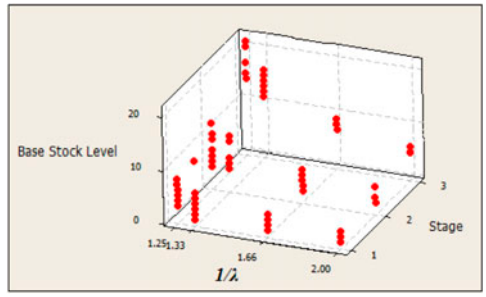


(a) KCS Policy

(b) EKCS Policy: Kanban

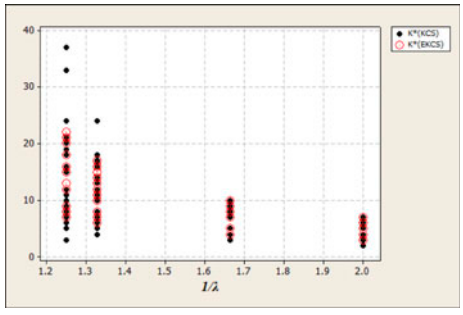


(c) BSCS Policy

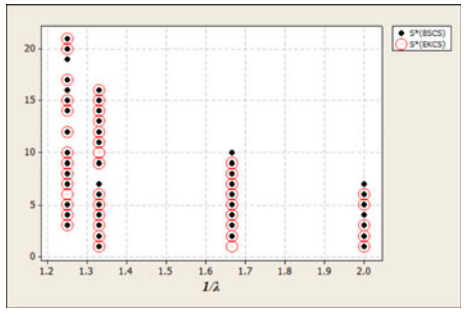


(d) EKCS Policy: Base Stock

Figure 4. Analysis of the approximate solution.



(a) Number of Kanban Tokens under KCS and EKCS policies



(b) Base Stock levels under KCS and EKCS policies

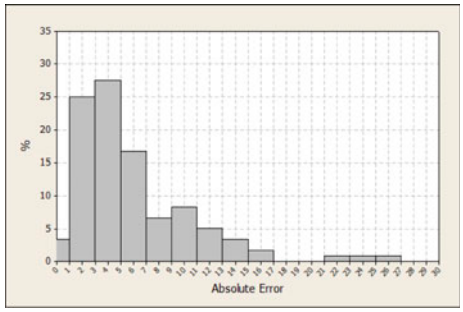
Figure 5. KCS and BSCS policies versus EKCS policy.

Figure 5: the optimal kanban obtained from running the EKCS model (empty circles in Figure 5(a)) is never larger than that obtained under the 'pure' KCS model (plain circles in Figure 5(a)). Moreover, in the case of small inter-arrival times, EKCS is characterised by a lower number of kanbans tokens because it can exploit the presence of base stock.

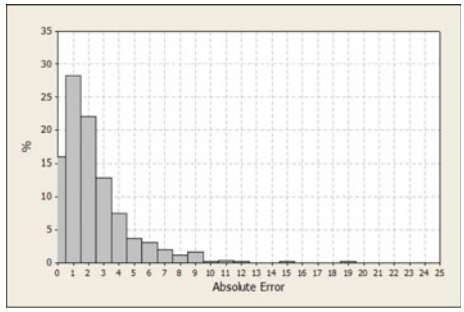
We can now investigate the difference between the approximate solution and the result from OptQuest©.

Figure 6 shows the histograms of the errors computed for all the experimental conditions. Figure 6(a) reports the absolute error for the 120 observations related to the objective function values and Figure 6(b) reports the absolute error on the 480 observations related to the components of the solution.

It can be observed that 75% of the observations related to the objective function value are characterised by an error less than 6 units (Figure 6(a)), whereas the error related to the solution components is below 3 units for the same percentage. Moreover, these aggregated graphs do not seem to show any multi-modality, i.e. the error seems to be concentrated around a unique mode. Nevertheless, we can notice a frequency increase for values of k far from the area where the density is concentrated ($k = \{24, \ldots, 27\}$ in Figure 6(a) and $k = \{15, 19\}$ in Figure 6(b)). This behaviour can be partially attributed to the sample size n, and it is mitigated as n increases. Indeed, low-frequency high-error classes, on the extreme right tail

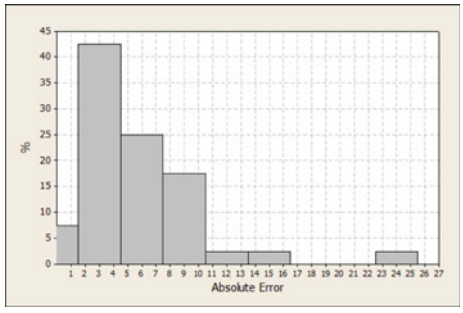


(a) Absolute error of the objective function value (120 observations)

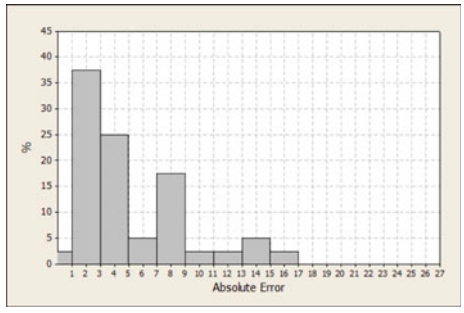


(b) Absolute error of each solution component (480 observations)

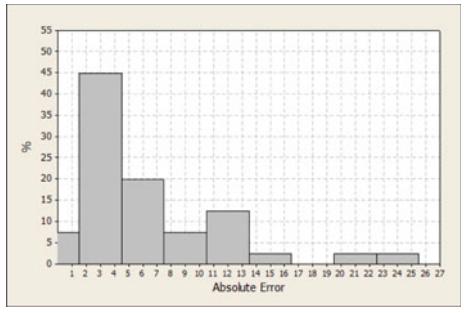
Figure 6. Absolute error analysis.



(a) KCS: Absolute Error computed for the objective function $|\kappa^* - \tilde{\kappa}|$



(b) BSCS: Absolute Error computed for the objective function $|\mathcal{B}^* - \tilde{\mathcal{B}}|$



(c) EKCS: Absolute Error computed for the objective function $|\mathcal{E}^* - \tilde{\mathcal{E}}|$

Figure 7. Absolute errors.

of the histogram, are the effect of the size n of the sample path. These phenomena are mitigated and eventually vanish as n increases.

Figures 7 report the error in the objective function. In the KCS case (Figure 7(a)), more than 45% of the observations are characterised by an error lower than 3 kanban units. Instead, in the BSCS case (Figure 7(b)), more than 45% of the observations are characterised by an error lower than 3 base stock units. Finally, more than 45% of the observations are characterised by an error lower than 4 units in the EKCS case (Figure 7(c)).

Analysing the results for the KCS policy (second and third columns in Table 1), we observe that the larger the arrival rate (i.e. the lower the inter-arrival time), the larger the error $\tilde{k}_j - k_j^*$ (both average e_μ and standard error $e_{\hat{\sigma}}$) computed as the difference between the number of kanban tokens resulting from OptQuest© and those from the proposed method. Specifically, when the inter-arrival time is large, $1/\lambda = 2.00$, i.e. twice the average processing time, we observed that 80% of the solutions obtained from independent replications were characterised by an individual error smaller than 2 kanban units, and the error density appeared substantially unimodal. On the contrary, under high-frequency demand, i.e. $1/\lambda = 1.25$ in

Table 1, corresponding to 80% of the processing time, we observed that only 35% of the generated solutions were characterised by an individual error smaller than 1 kanban unit, and the error density was apparently multi-modal.

In Table 1, we observe the effect of the demand rate of positively shifting the location of the error distribution meanwhile increasing the variance. When the system is highly saturated, the length of the simulation–optimisation sample path n becomes a critical parameter because we need more observations for the model to converge to the optimal solution, and if n is not sufficient, the solution is biased by the transient period effect (Pedrielli 2013).

Nevertheless, as the sample size increases, these effects are mitigated and eventually vanish (Pedrielli 2013). Pilot experiments were run with $n > 10,000$ showing a substantial reduction of the standard error (concerning the KCS, experiments with 10, 000 jobs with $1/\lambda = 1.25$ led to a standard error $e_{\hat{\sigma}_{j\,1}} = 0.532$, $e_{\hat{\sigma}_{j\,2}} = 0.703$, $e_{\hat{\sigma}_{j\,3}} = 0.726$). It is also important to consider that the observations refer to absolute values of the error: as the system saturates, the amount of kanban tokens needed increases, so the absolute error increases while the relative error is preserved.

Table 1 also shows the effect of the stage j on the error. It is possible to observe that the last stage $j = 3$ is the most critical. Nevertheless, we observed that 50% of the generated solutions showed an error smaller than 4 kanban units.

Under the BSCS policy, as in the KCS case, when the demand rate is low, i.e. $1/\lambda = 2.00$, we observed 75% of the generated solutions were characterised by an error of less than 2 basestock units, and the estimated error density function was unimodal. Under frequent demand, i.e. $1/\lambda = 1.25$, we observed less than 35% of the generated solutions were characterised by an error smaller than 2 basestock units, and the estimated error density showed multi-modality. The effect of the sample size n is the same as for the KCS. Observing the EKCS results, we notice that, as in the BSCS case, a better approximation of the exact solution is obtained at the demand stage ($j = 3$) with respect to the KCS and the performance results even improved with respect to the BSCS policy. Also, an increased error in the first stage is observed. Apparently, the first stage results are critical because of the conflict between the KCS policy that tends to delay the release of the job to the system and the BSCS that wishes to maximise the service level. This mixed effect is hard to approximate.

It is important to outline the effect of the stage over the approximate model error in the three policies. KCS shows an increased error in the last stage and the error is exacerbated when the demand rate increases. This effect is mitigated in the BSCS and we observed that, in more than 65% of the cases, the model returns an absolute error less than 2 base stock units on the last stage. This improved performance is due to the demand signal that is received by all stages. Indeed, the same improvement is noticed when considering the EKCS policy. A further intuition behind the better approximation performance in terms of the obtained solution of the time buffer model when applied to BSCS and EKCS resides in the similarities between the time-based and space-based inventory model. In the case of the base stock problem (and, in part, the EKCS model), the time buffer can be interpreted as the desired coverage interval, i.e. the time for which the WIP would be able to 'cover' the demand before stock out, if no additional item is processed. This is similar to delay the start of the production in previous stages when demand is received at the last. The Kanban system does not show such a similarity with the time-based model and the approximation results generally poorer.

Table 2 reports the error, in terms of both mean and standard error, obtained running 10 independent replications for the KCS policy with inter-arrival time $1/\lambda = 1.665$ and varying the number of stages from $j = 4$ to $j = 7$. From Table 2, we can observe that the error is not remarkably influenced by the increased complexity of the system in terms of number of stages. As a result, we can infer that the main factor affecting the performance of the proposed approach is the traffic intensity. As the arrival rate increases, the error increases as well especially at the last stage, that is, the one subject to the system demand. On the contrary, the size of the problem does not influence the quality of the solution.

6.2.1 *Summary results*

Here, we summarise the main findings resulting from the presented experimentation:

- The approximation error increases in terms of both mean and standard error for all the policies as the demand rate increases (inter-arrival time decreases);
- The analysis of the error frequencies showed a multi-modal behaviour under high-frequency demand, revealing the need to increase the number of samples n;
- The approximation of the KCS is poor at the last stage of the line; this effect is mitigated in the BSCS as well as in the EKCS. Nonetheless, EKCS shows an increased error on the first stage;
- The error does not increase as the number of stages increase, but it keeps showing the same pattern (i.e. larger error in the source and sink stages);
- The variance of the error decreases as the number of jobs increases.

Table 1. Approximate solution error (mean and standard error) decomposition by stage j and inter-arrival time $1/\lambda$.

Stage		KCS		BSCS		EKCS	
Policy							
$\lvert x_j - x_j^* \rvert$		e_μ	$e_{\hat{\sigma}}$	e_μ	$e_{\hat{\sigma}}$	e_μ	$e_{\hat{\sigma}}$
$j = 1$							
$1/\lambda = 2.00$		1.4	0.163	1.1	0.233	1.7	0.141
$1/\lambda = 1.66$		2.338	0.241	0.7	0.260	2.45	0.171
$1/\lambda = 1.33$		1.942	0.441	3	0.558	2.4	0.515
$1/\lambda = 1.25$		1.930	0.685	2	0.577	6.35	0.703
$j = 2$							
$1/\lambda = 2.00$		0.9	0.179	0.8	0.2	0.9	0.113
$1/\lambda = 1.66$		0.8	0.512	2.1	0.433	1.9	0.201
$1/\lambda = 1.33$		2.78	0.824	2.3	0.803	2.55	0.360
$1/\lambda = 1.25$		4.78	0.940	2.5	0.522	5	0.553
$j = 3$							
$1/\lambda = 2.00$		0.2	0.133	1.8	0.2	0.9	0.115
$1/\lambda = 1.66$		3.25	0.638	1	0.211	1.3	0.176
$1/\lambda = 1.33$		3.4	1.614	1.5	0.269	1.5	0.241523
$1/\lambda = 1.25$		9.41	1.643	5.2	0.892	2.6	0.307

Table 2. KCS: error in the approximate solution with $j = \{4, 5, 6, 7\}$ and inter-arrival time $1/\lambda = 1.665$.

Stage	$J = 4$		$J = 5$		$J = 6$		$J = 7$	
	e_μ	$e_{\hat{\sigma}}$	e_μ	$e_{\hat{\sigma}}$	e_μ	$e_{\hat{\sigma}}$	e_μ	$e_{\hat{\sigma}}$
$j = 1$	1.500	0.269	1.500	0.224	1.200	0.200	1.400	0.267
$j = 2$	1.700	0.335	0.400	0.163	0.400	0.267	1.200	0.327
$j = 3$	1.500	0.269	1.800	0.249	0.700	0.260	2.500	0.269
$j = 4$	1.400	0.427	1.400	0.581	1.400	0.306	0.800	0.200
$j = 5$	–	–	3.200	0.593	1.600	0.371	0.900	0.233
$j = 6$	–	–	–	–	1.000	0.516	1.000	0.577
$j = 7$	–	–	–	–	–	–	1.500	0.307

6.3 *Benchmark models: results*

In this section, we show two examples of benchmark models (Section 5.4). In particular, we refer to two performance functions and, consequently, to two benchmark models P_1 and P_2.

$$P_1 : \sum_{j=1}^{J} \sum_{i=1}^{n} c_j \left(z_{i,j+1} - z_{i,j} \right) \tag{33}$$

Objective (33) of problem P_1 is the one used as example in Section 5.4 (Equation (27)) to minimise the costs. In particular, in these experiments, we assumed costs increasing in the system stage, i.e., $\mathbf{c} = \{10, 15, 20\}$.

$$P_2 : \sum_{i=1}^{n} \left(z_{i,J+1} - z_{i,1} \right) \tag{34}$$

Problem P_2 does not consider any cost and minimises the total waiting time in the system. We compare the performance obtained from the time buffer models (KCS, BSCS and EKCS) with that obtained from the benchmark models. In particular, we look at the following performance measures for all the models:

- Average service level, $\vartheta_1 = \left(1 - \frac{1}{n} \sum_{i=1}^{n} 1_{(z_{i,J+1} - d_i > 0)} \right)$ defined as the portion of customers satisfied with delay over the total number of customers arriving at the system (no customer loss is allowed);

Table 3. Benchmark models, mean and standard error of the performance.

Policy	ϑ_1	$\hat{\sigma_1}$	ϑ_2	$\hat{\sigma_2}$	ϑ_3	$\hat{\sigma_3}$	ϑ_4	$\hat{\sigma_4}$
P_1	0.923	<0.04	7.540	0.383	765008	28742	8.413	0.343
P_2	0.926	<0.04	7.452	0.380	887703	35077	8.381	0.342
BSCS	0.964	<0.04	29.483	1.886	2659419	154222	25.956	1.592
KCS	0.965	<0.04	29.924	1.834	2731266	157627	26.319	1.548
EKCS	0.964	<0.04	28.394	1.864	2590358	153838	25.084	1.570

- Average waiting time, $\vartheta_2 = \frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n}\left(z_{i,j+1} - y_{i,j} + x_{i,j} - z_{i,j}\right) = \frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n}\left(z_{i,j+1} - z_{i,j} - t_{ij}\right)$, i.e. the time between the start of processing at server j and the release to the jth stage plus the time between the release to the $j+1$th stage and the end of the processing at server j;
- Average cost, $\vartheta_3 = \frac{1}{n}\sum_{j=1}^{J}\sum_{i=1}^{n} c_j\left(z_{i,j+1} - z_{i,j}\right)$;
- Average queue at each stage ϑ_4.

We consider the same experimental settings adopted in previous experiments, but we used a single inter-arrival time $1/\lambda = 1.25$ throughout the experimentation. We compare the performance of the KCS, BSCS and EKCS against the benchmark policies.

Table 3 shows the obtained results with the standard error. The results in Table 3 suggest that the extended kanban control policy is the best policy in terms of waiting time, whereas the KCS results the most expensive as it yields the largest waiting time. This outcome confirms the results obtained in Geraghty and Heavey (2005) and Frein, Di Mascolo, and Dallery (1995). The KCS provides the worst performance because of the delay in the demand transmission. The low performance is confirmed when the WIP (ϑ_4 in Table 3) is considered.

As Table 3 reveals, an important role played by the benchmark models is to quantify the *lower bound* on the achievable performance. Indeed, observing the service level (ϑ_1), we notice that both policies P_1 and P_2 are worse than BSCS, KCS and EKCS. In fact, the benchmark policies P_1 and P_2 return the worst performance and the difference with BSCS, KCS and EKCS is statistically significant (95% confidence). This is due to the fact that both P_1 and P_2 minimise the waiting time, hence, they try to delay the release of jobs to stages in order to avoid waiting. Nonetheless, this imposed delay reflects in a reduced capacity to timely respond to demand surges. We can then state that the proposed policy should not return a service level lower than the one provided by the benchmarks.

7. Conclusions

This paper proposes new MIPs models for the integrated simulation–optimisation of pull control systems for optimising the parameters of KCS, BSCS and EKCS policies. In order to solve the models, the time buffer approximation is applied to transform the MIP into their linear counterpart that can be easily solved using commercial optimisers.

The approximate models provide as output the optimal time buffer allocation, the approximate integer solution to the original MIP problems, and the performance estimates of the system under the computed time buffer solution. Numerical experiments show that the approximate solution is a high-quality solution because it is always close to the real optimum.

Benchmark models were introduced for the first time as a mean to evaluate the quality of the proposed policy with respect to the best and worst achievable performances.

Several directions for future research are being investigated by the authors. In particular, further modelling enhancements are under analysis in order to define the complexity of the layout that can be considered through the presented methodologies. Furthermore, the design of time buffer based policies is under study. Following this direction, the time buffer does not represent the approximation of a discrete entity, but the parameter of a time-based policy. The performance and the ease of implementation of such a policy are both current research subjects. Finally, the development of benchmark models is a topic of interest. In particular, the authors are investigating several relaxations of the original models in order to provide tighter bounds on the performance.

Acknowledgements

The authors wish to thank each of the three anonymous reviewers for their fruitful comments which substantially contributed to the improvement of the paper.

Funding

This work was partially supported by the National Science Foundation of China under Grant 61473188 and Singapore Maritime Institute under Grant R-SMI-2013-MA-11.

References

Aglan, Canan, and Mehmet Bulent Durmusoglu. 2014. "Lot-splitting Approach of a Hybrid Manufacturing System Under CONWIP Production Control: A Mathematical Model." *International Journal of Production Research* (ahead-of-print) :1–23. In press.

Alfieri, A., and A. Matta. 2012a. "Mathematical Programming Formulations for Approximate Simulation of Multistage Production Systems." *European Journal of Operational Research* 219 (3): 773–783.

Alfieri, A., and A. Matta. 2012b. "Mathematical Programming Representation of Pull Controlled Single-product Serial Manufacturing Systems." *Journal of Intelligent Manufacturing* 23 (1): 23–35.

Alfieri, A., and A. Matta. 2013. "Mathematical Programming Time-based Decomposition Algorithm for Discrete Event Simulation." *European Journal of Operational Research* 231 (3): 557–566.

Alfieri, A., A. Matta, and G. Pedrielli. In press. "Mathematical Programming Formulations for Approximate Simulation Optimization of Closed-loop Systems." *Annals of Operations Research*. doi: 10.1007/s10479-013-1480-7.

Axster, S., and K. Rosling. 1993. "Installation vs. Echelon Stock Policies for Multilevel Inventory Control." *Management Science* 39 (10): 1274–1280.

Bonvik, A. M., C. E. Couch, and S. B. Gershwin. 1997. "A Comparison of Production-line Control Mechanisms." *International Journal of Production Research* 35 (3): 789–804.

Chan, W. K., and L. W. Schruben. 2003. "Properties of Discrete Event Systems from Their Mathematical Programming Representations." In *Proceedings of the 2003 Winter Simulation Conference*, edited by S. Chick, P. J. Sanchez, D. Ferrin, and D. J. Morrice, 496–502. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Chan, W. K., and L. W. Schruben. 2008a. "Mathematical Programming Models of Closed Tandem Queueing Networks." *ACM Transactions on Modeling and Computer Simulation* 19 (1).

Chan, W. K., and L. W. Schruben. 2008b. "Optimization Models of Discrete-event System Dynamics." *Operations Research* 56 (5): 1218-1237.

Cochran, J. K., and H. A. Kaylani. 2008. "Optimal Design of A Hybrid Push/Pull Serial Manufacturing System with Multiple Part Types." *International Journal of Production Research* 46 (4): 949–965.

Cottenceau, B., L. Hardouin, and I. Ouerghi. 2008. "Kanban Policy Improvement Thanks to a (max, +)-Algebra Analysis." *International Journal of Systems Science* 39 (7): 689–698.

Dallery, Y., and G. Liberopoulos. 2000. "Extended Kanban Control System: Combining Kanban and Base Stock." *IIE Transactions* 32 (4): 369–386.

Daniel, J. S. R., and C. Rajendran. 2006. "Heuristic Approaches to Determine Base-stock Levels in a Serial Supply Chain with a Single Objective and with Multiple Objectives." *European Journal of Operational Research* 175 (1): 566–592.

Deleersnyder, J.-L., T. J. Hodgson, H. Muller-Malek, and P. J. O'Grady. 1989. "Kanban Controlled Pull Systems: An Analytic Approach." *Management Science* 35 (9): 1079–1091.

Di Mascolo, M., Y. Frein, and Y. Dallery. 1996. "An Analytical Method for Performance Evaluation of Kanban Controlled Production Systems." *Operations Research* 44 (1): 50–64.

Duri, C., Y. Frein, and M. Di Mascolo. 2000. "Comparison Among Three Pull Control Policies: Kanban, Base Stock, and Generalized Kanban." *Annals of Operations Research* 93 (1–4): 41–69.

Frein, Y., M. Di Mascolo, and Y. Dallery. 1995. "On the Design of Generalized Kanban Control Systems." *International Journal of Operations and Production Management* 15 (9): 158–184.

Fu, M. 2002. "Optimization for Simulation: Theory vs. Practice." *Journal on Computing* 14 (3): 192–215.

Gaury, E. G. A., J. P. C. Kleijnen, and H. Pierreval. 2001. "A Methodology to Customize Pull Control Systems." *Journal of the Operational Research Society* 52 (7): 789–799.

Gaury, E. G. A., H. Pierreval, and J. P. C. Kleijnen. 2000. "An Evolutionary Approach to Select a Pull System Among Kanban, Conwip and Hybrid." *Journal of Intelligent Manufacturing* 11 (2): 157–167.

Geraghty, J., and C. Heavey. 2005. "A Review and Comparison of Hybrid and Pull-type Production Control Strategies." *OR Spectrum* 27 (2–3): 435–457.

Hao, Q., and W. Shen. 2008. "Implementing a Hybrid Simulation Model for a Kanban-based Material Handling System." *Robotics and Computer-Integrated Manufacturing* 24 (5): 635–646.

Helber, S., K. Schimmelpfeng, and R. Stolletz. 2011. "Setting Inventory Levels of CONWIP Flow Lines via Linear Programming." *BuR Business Research Journal* 4 (1): 98–115.

Helber, S., K. Schimmelpfeng, R. Stolletz, and S. Lagershausen. 2011. "Using Linear Programming to Analyze and Optimize Stochastic Flow Lines." *Annals of Operations Research* 182 (1): 193–211.

Hopp, W. J., and M. L. Spearman. 2007. *Factory Physics*. New York: McGraw-Hill.

Huang, C.-C., and A. Kusiak. 1996. "Overview of Kanban Systems." *International Journal of Computer Integrated Manufacturing* 9 (3): 169–189.

Karaesmen, F., and Y. Dallery. 2000. "A Performance Comparison of Pull Type Control Mechanisms for Multi-Stage Manufacturing." *International Journal of Production Economics* 68 (1): 59–71.
Kleijnen, J. P. C. 2008. "Design and Analysis of Simulation Experiments." Vol. 111. *International Series in Operations Research & Management Science*. New York: Springer.
Köchel, P., and U. Nieländer. 2002. "Kanban Optimization by Simulation and Evolution." *Production Planning & Control* 13 (8): 725–734.
Koulouriotis, D. E., A. S. Xanthopoulos, and V. D. Tourassis. 2010. "Simulation Optimisation of Pull Control Policies for Serial Manufacturing Lines and Assembly Manufacturing Systems Using Genetic Algorithms." *International Journal of Production Research* 48 (10): 2887–2912.
Liberopoulos, G., and Y. Dallery. 2000. "A Unified Framework for Pull Control Mechanisms in Multi-stage Manufacturing Systems." *Annals of Operations Research* 93 (1–4): 325–355.
Matta, A. 2008. "Simulation Optimization with Mathematical Programming Representation Of Discrete Event Systems." In *Proceedings of the 2008 Winter Simulation Conference*, edited by S. J. Mason, R. R. Hill, L. Monch, O. Rose, T. Jefferson, and J. W. Fowler, 1393–1400. Piscataway, NJ: Institute of Electrical and Electronics Engineers Inc.
Matta, A., Y. Dallery, and M. Di Mascolo. 2005. "Analysis of Assembly Systems Controlled with Kanbans." *European Journal of Operational Research* 166 (2): 310–336.
Ohno, K. 2011. "The Optimal Control of Just-in-time-based Production and Distribution Systems and Performance Comparisons with Optimized Pull Systems." *European Journal of Operational Research* 213 (1): 124–133.
Oladipupo, O., A. Rotondo, P. Young, and J. Geraghty. 2014. "On Designing Robust Kanban Production Control Strategies in Multiproduct Manufacturing Environments." Vol. 68. In *Handbook of Research on Design and Management of Lean Production Systems*, edited by V. Modrák and P. Semano, 68–88. Košice: IGI Global.
Park, Chan-Woo, and Hyo-Seong Lee. 2013. "Performance Evaluation of a Multi-product CONWIP Assembly System with Correlated External Demands." *International Journal of Production Economics* 144 (1): 334–344.
Pedrielli, G. 2013. "Discrete Event Systems Simulation-Optimization: Time Buffer Framework." PhD thesis. Mechanical Engineering Department, Politecnico di Milano, Italy.
Rouwenhorst, B., B. Reuter, V. Stockrahm, G. J. van Houtum, R. J. Mantel, and W. H. M. Zijm. 2000. "Warehouse Design and Control: Framework and Literature Review." *European Journal of Operational Research* 122 (3): 515–533.
Schruben, LW. 2000. "Mathematical Programming Models of Discrete Event System Dynamics." In *Proceedings of the 2000 Winter Simulation Conference*, edited by J. A. Joines, R. R. Bartona, K. Kang, and P. A. Fishwick, 381–385. Piscataway, NJ: Institute of Electrical and Electronics Engineers.
Schwarz, J.A., and R. Stolletz. 2013. "A Sampling Approach for the Analysis of Time-dependent Stochastic Flow Lines." In *Proceedings of the 2013 Conference on Stochastic Models of Manufacturing and Service Operations*, Kloster Seeon, Germany.
Shang, K. H., and J.-S. Song. 2007. "Serial Supply Chains with Economies of Scale: Bounds and Approximations." *Operations Research* 55 (5): 843–853.
Shang, K. H., J.-S. Song, and P. H. Zipkin. 2009. "Coordination Mechanisms in Decentralized Serial Inventory Systems with Batch Ordering." *Management Science* 55 (4): 685–695.
Smew, W., P. Young, and J. Geraghty. 2013. "Supply Chain Analysis Using Simulation, Gaussian Process Modelling and Optimisation." *Operational Research* 186 (3): 1099–1113.
Song, D.-P. 2013. "Threshold-type Control Policies and System Stability for Serial Supply Chain Systems." In *Optimal Control and Optimization of Stochastic Supply Chain Systems*, 131–148. New York: Springer.
Spearman, M. L., and M. A. Zazanis. 1992. "Push and Pull Production Systems: Issues and Comparisons." *Operations Research* 40 (3): 521–532.
Stolletz, R, and S. Weiss. 2013. "Buffer Allocation Using Exact Linear Programming Formulations and Sampling Approaches." In *Preprints of the 2013 IFAC Conference on Manufacturing Modelling, Management, and Control*, Saint Petersburg, Russia.
Wang, L.-F., and L.-Y. Shi. 2013. "Simulation Optimization: A Review on Theory and Applications." *Acta Automatica Sinica* 39 (11): 1957–1968.
Yan, H., G. Yin, and S. X. C. Lou. 1994. "Using Stochastic Optimization to Determine Threshold Values for the Control of Unreliable Manufacturing Systems." *Journal of Optimization Theory and Applications* 83 (3): 511–539.
Yang, T., H.-P. Fu, and K.-Y. Yang. 2007. "An Evolutionary-simulation Approach for the Optimization of Multi-constant Work-in-process Strategya case study." *International Journal of Production Economics* 107 (1): 104–114.

Appendix 1. Proofs

Proof Proof of Property 1. Consider constraints (12). Job i and job $i + k$ are linked by the following constraint:

$$z_{i+k,j} \geq z_{i,j+1} - s^K_{kj}.$$

The release time of job i from stage $j + 1$, $z_{i,j+1}$ also appears in the constraint linking job i and job $i + k + 1$:

$$z_{i+k+1,j} \geq z_{i,j+1} - s^K_{k+1,j}.$$

Since job $i+k$ is served before customer $i+k+1$, the following condition must hold:

$$z_{i+k,j} \leq z_{i+k+1,j}.$$

If $s^K_{k,j} \leq s^K_{k+1,j}$, it would allow job $i+k+1$ to be released to stage j before job $i+k$. However, since job $i+k$ must precede customer $i+k+1$, the additional time $s^K_{k+1,j} - s^K_{k,j}$ is not used by job $i+k+1$. Hence, there always exists an optimal solution such that $s^K_{k,j} \geq s^K_{k+1,j}$.

Proof Proof of Property 2. The primal (on the left) and the dual (on the right) approximate optimisation models, in their matrix forms, can be formulated as follows.

$$\begin{aligned} &\min \quad \mathcal{S} = \mathbf{1}'\mathbf{s} && \max \quad \mathbf{r}'\mathbf{u}' + \mathbf{d}'\mathbf{u}'_\zeta - \left(\alpha^* + n \cdot \mathbf{1}' \cdot \mathbf{d}\right) \cdot \theta \\ &\text{s.t.} \\ &\mathbf{A^1}\upsilon \geq \mathbf{r}\ (\mathbf{u}) && \mathbf{A^{1'}}\mathbf{u} \leq [0|\mathbf{1}] \\ &\mathbf{A^2}\zeta \geq \mathbf{d}\ (\mathbf{u}_\zeta) && \mathbf{A^{2'}}\mathbf{u}_\zeta \leq 0 \\ &\frac{1}{n}\sum_{i=1}^{n}\left(z_{i,J+1} - d_i\right) \leq \alpha^*\ (\vartheta) && \vartheta \geq 0 \end{aligned}$$

The vector υ' is the set of variables of the primal model, i.e. the time buffer $\mathbf{s}$ and the event times ($\mathbf{z}$, $\mathbf{x}$, $\mathbf{y}$), while $\mathbf{u}'$, $\mathbf{u}'_\zeta$ and θ represent the set of dual variables; in particular, $\mathbf{u}'_\zeta$ is the set of dual variables referring to constraints (6).

The matrix $\mathbf{A} = \left[A^1|A^2\right]$ is an $l \times m$ dimension matrix, where: 1) $l = (n-1)J + 2 \cdot J + n + n \cdot J + \sum_{j=1}^{J}\sum_{k_j=1}^{k_j up}(n-k)$ represents the number of constraints not including the performance constraint (8); 2) $m = 3nJ + n + \sum_{j=1}^{J} k_j^{up}$ represents the number of decision variables.

The m-dimensional vector of the right-hand side $\mathbf{r} = \{r_1, r_2, \ldots, r_m\}$ consists of the realisation of the random variables representing the processing times t_{ij} and the customers inter-arrival times d_i.

The objective of the primal problem, $\mathcal{S}$, is a function of the time buffer $\mathbf{s}$, which depends on the right hand side $\mathbf{r}$, the target performance α^* and the inter-arrival times $\mathbf{d}$.

Two target performance are given, α_1^* and α_2^*, such that $\alpha_1^* \geq \alpha_2^*$. Then $\mathcal{S}^*$, at the optimum, satisfies:

$$\begin{aligned} \mathcal{S}^*(\alpha_1^*) &= \mathbf{r}'\mathbf{u}^{(1)*} + \mathbf{d}'\mathbf{u}_\zeta^{(1)*} - \left(\alpha_1^* + \mathbf{d}\right)\theta^{(1)*} \\ &\leq \mathbf{r}'\mathbf{u}^{(1)*} + \mathbf{d}'\mathbf{u}_\zeta^{(1)*} - \left(\alpha_2^* + \mathbf{d}\right)\theta^{(1)*} \\ &\leq \mathbf{r}'\mathbf{u}^{(2)*} + \mathbf{d}'\mathbf{u}_\zeta^{(2)*} - \left(\alpha_2^* + \mathbf{d}\right)\theta^{(2)*} = \mathcal{S}^*(\alpha_2^*), \end{aligned}$$

where $\left(\mathbf{u}^{(1)*}, \mathbf{u}_\zeta^{(1)*}, \theta^{(1)*}\right)$ and $\left(\mathbf{u}^{(2)*}, \mathbf{u}_\zeta^{(2)*}, \theta^{(2)*}\right)$ are optimal solutions of the dual problems having as right hand side $\left(\mathbf{r}, \alpha_1^*, \mathbf{d}\right)$ and $\left(\mathbf{r}, \alpha_2^*, \mathbf{d}\right)$, respectively. This proves $\mathcal{S}$ is non-increasing in α^*.

Proof Proof of Property 3. Consider constraints (20). Job i and job $i+k$ are related as follows:

$$z_{i+k,j} \geq d_i - s^e_{kj}.$$

The arrival time of customer i, d_i also appears in the constraint linking job i and job $i+k+1$:

$$z_{i+k+1,j} \geq d_i - s^e_{k+1,j}.$$

Since job $i+k$ is served before customer $i+k+1$, the following condition must hold:

$$z_{i+k,j} \leq z_{i+k+1,j}.$$

If $s^e_{k,j} \leq s^e_{k+1,j}$, it would allow job $i+k+1$ to be released to stage j before job $i+k$. However, since job $i+k$ must precede customer $i+k+1$, the additional time $s^e_{k+1,j} - s^e_{k,j}$ is not used by job $i+k+1$. Hence, there always exists an optimal solution such that $s^e_{k,j} \geq s^e_{k+1,j}$.

Proof Proof of Property 4. The proof of the Property 4 is the same as the proof for Property 2, therefore it is not reported here.

Proof Proof of Property 5 Consider constraints (12) and (20), we can rewrite them as follows (only for the last stage):

$$\begin{aligned} z_{i+k,J} &\geq z_{i,J+1} - s^K_{kJ}, \\ z_{i+k,J} &\geq d_i - s^e_{kJ}. \end{aligned}$$

Since a unique model is run to compute the optimal time buffer for both kanban and base stock, both constraints refer to the same value of $z_{i+k,J}$. We know that $z_{i,J+1} \geq d_i$ for constraints (6). As a result $s^K_{kJ} \geq s^e_{kJ}$.