Optimal Buffer Allocation in Serial Production Lines Operating under IB, EB, and CONWIP Policies

George LIBEROPOULOS
Department of Mechanical Engineering, University of Thessaly, Volos, Greece, email: glib@mie.uth.gr

We consider a production line consisting of several machines in series with finite intermediate buffers. The machines have geometrically distributed processing times. We address the problem of determining the optimal buffer sizes to maximize the average profit of the line subject to a minimum average throughput constraint, under three operating policies. The average profit is defined as the weighted average throughput of the line minus the sum of the weighted average WIP plus total buffer capacity. The considered policies are: installation buffer (IB), echelon buffer (EB), and CONWIP. IB is the traditional policy under which a machine can store the parts that it produces only in its immediate downstream buffer. Under EB, it can store them in any of its downstream buffers. CONWIP is a special case of EB where the capacities of all buffers, except the last one, are zero. To find the optimal buffer allocation for each policy, we use a two-step gradient algorithm, where the average profit for given buffer sizes is estimated using decomposition-based approximation. Numerical results show that the optimal EB policy outperforms the optimal IB and CONWIP policies.

Keywords: production line; installation buffer; echelon buffer; CONWIP; buffer allocation

1 Introduction

We consider a production line consisting of N machines in series denoted by $M_n, n = 1, \dots, N$, with $N-1$ finite intermediate buffers denoted by B_n, $n = 1, \dots, N-1$. Time is broken in discrete periods. In each period, machine $M_n, n = 1, \dots, N$, produces a part with probability p_n unless it is starved or blocked; hence, the processing time of a part on machine M_n is geometrically distributed with mean $1/p_n$. Probability p_n is referred to as the production probability (or rate) of machine M_n in isolation. Each machine can hold one part. The capacity of buffer B_n is denoted by $C_n, n = 1, \dots, N-1$.

In the traditional way of operating such a line, machine $M_n, n = 1, \dots, N-1$, is allowed to store the parts that it produces only in its immediate downstream buffer B_n if the next machine M_{n+1} is occupied. We refer to the ensemble of B_n and M_{n+1} as the installation buffer following M_n and we denote it by I_n, i.e., $I_n = B_n \cup M_{n+1}$. Clearly, the capacity of I_n is $1 + C_n$. Moreover, we denote the number of parts in I_n by $i_n, n = 1, \dots, N-1$. We refer to i_n as the installation WIP following machine M_n and to the resulting way of operation as installation buffer (IB) policy. Under the IB policy, machine M_n is blocked (before service) from processing a part if the number of parts that have been produced by it but have not yet departed from the next machine M_{n+1} is equal to the capacity of I_n, i.e., if $i_n = 1 + C_n, n = 1, \dots, N-1$. Figure 1 shows a production line with $N = 4$ machines and $N - 1 = 3$ intermediate buffers operated under IB.

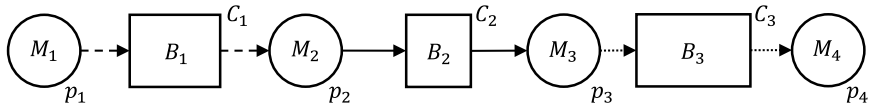

Figure 1. Serial production line operated under an IB policy.

Inserting buffers between machines comes at a cost of additional WIP inventory, capital investment, and floor space. Depending on the industry, such a cost can be quite high. The optimal allocation of storage capacity among the intermediate buffers is one of the most widely studied problems in manufacturing systems research. Even if the total capacity has been optimized, storing parts locally in the intermediate buffers does not take full advantage of this capacity. When the cost of buffer space is significant, it may be worthwhile to consider increasing the utilization of the existing buffers before setting out to increase total buffer capacity.

Recently, Liberopoulos (2017) proposed an operating policy that aims to increase the utilization of buffer space by allowing machine $M_n, n = 1, \dots, N-1$, to store the parts that it produces in any of its downstream buffers, $B_n, \dots, B_{N-1}$, if the next machine M_{n+1} is occupied. We refer to the ensemble of $B_n, \dots, B_{N-1}$, and

M_{n+1} as the echelon buffer following M_n and we denote it by E_n, i.e., $E_n = B_n \cup \cdots \cup B_{N-1} \cup M_{n+1}$. Clearly, the capacity of E_n is $1 + \sum_{m=n}^{N-1} C_m, n = 1, \ldots, N-1$. Moreover, we denote the number of parts in E_n by e_n, $n = 1, \ldots, N-1$. We refer to e_n as the echelon WIP following machine M_n and to the resulting way of operation as echelon buffer (EB) policy. Under the EB policy, machine M_n is blocked (before service) from processing a part if the number of parts that have been produced by it but have not yet departed from the last machine M_{N-1} is equal to the capacity of E_n, i.e., if $e_n = 1 + \sum_{m=n}^{N-1} C_m, n = 1, \ldots, N-1$. From the point of view of buffers, under the EB policy, each buffer is shared by all its upstream machines. Figure 2 shows the same production line as that in Figure 1 operated under an EB policy.

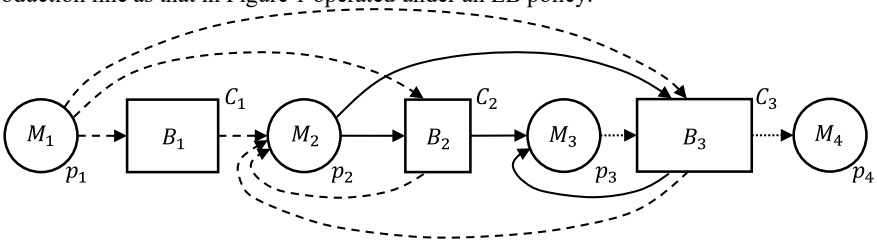

Figure 2. Serial production line operated under an EB policy.

If the capacities of all intermediate buffers, except possibly the last one, are zero (i.e., if $C_n = 0, n = 1, \ldots, N-2$, and $C_{N-1} \geq 0$), then under the EB policy, machine $M_n, n = 1, \ldots, N-1$, can store the parts that it produces in the last and only buffer B_{N-1} if M_{n+1} is occupied. To simplify notation, we denote this buffer by B and its capacity by C, i.e., $B \equiv B_{N-1}$ and $C = C_{N-1}$. In this case, it is easy to see that M_1 is blocked from processing a part if $e_1 = 1 + C$ and that no other machine is ever be blocked. This way of operation is identical to the operation of CONWIP where parts are not allowed to be released into the system if the total WIP is at the WIP-cap (Spearman et *al*. 1990). For the purposes of this paper, we will henceforth refer to an EB policy where all buffers except the last one have zero capacities and the last buffer has capacity $C \geq 0$, as CONWIP with WIP-cap $1 + C$. Figure 3 depicts the production line of Figures 1 and 2 operated under CONWIP, where the last buffer is shown as a common storage area on the side of the machines.

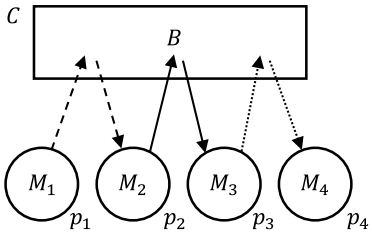

Figure 3. Serial production line operated under a CONWIP policy.

To analyze the operation of a production line under the EB policy, Liberopoulos (2017) modelled it as a token-based queueing network and developed a decomposition-based approximation method for evaluating its performance. This queuing network model consists of the N machines of the line, $M_1, \ldots, M_N$, separated by $N-1$ infinite capacity buffers, denoted by $Y_1, \ldots, Y_{N-1}$, as shown in Figure 4, for $N = 4$. The number of parts in buffer $Y_n, n = 1, \ldots, N-1$, including the part in machine M_{n+1} is denoted by y_n and is referred to as the stage WIP following M_n; y_n represents the number of parts that have been produced by M_n but have not yet departed from M_{n+1}. In the physical system shown in Figure 2, these parts may reside anywhere in $B_n \cup \cdots \cup B_{N-1} \cup M_{n+1}$. When a part flows from machine M_n to buffer Y_n, a token is generated and is placed in an associated finite buffer denoted by $E_n, n = 1, \ldots, N-1$. The number of tokens in $E_n, n = 1, \ldots, N-1$, is denoted by e_n and represents the echelon WIP downstream of M_n, i.e., the number of parts that have been produced by machine M_n but have not yet departed from the line. Hence, E_n is a surrogate of the echelon buffer following M_n (i.e., the ensemble $B_n \cup \cdots \cup B_{N-1} \cup M_{n+1}$) in the physical line shown in Figure 2.

The vertical line at the end of the system in Figure 4 represents an assembly operation that merges parts exiting the line with tokens from the echelon buffers. Thus, when a part is produced by machine M_N, it draws a token from each of the echelon buffers $E_1, \ldots, E_{N-1}$, signaling that all echelon WIP levels have dropped by one unit. The finished part leaves the line, and the tokens are discarded. The echelon WIP levels and the stage

WIP levels are related as follows: $e_n = \sum_{m=n}^{N-1} y_m, n = 1, \dots, N-1$; alternatively, $y_n = e_n - e_{n+1} \geq 0, n = 1, \dots, N-2$ and $y_{N-1} = e_{N-1}$.

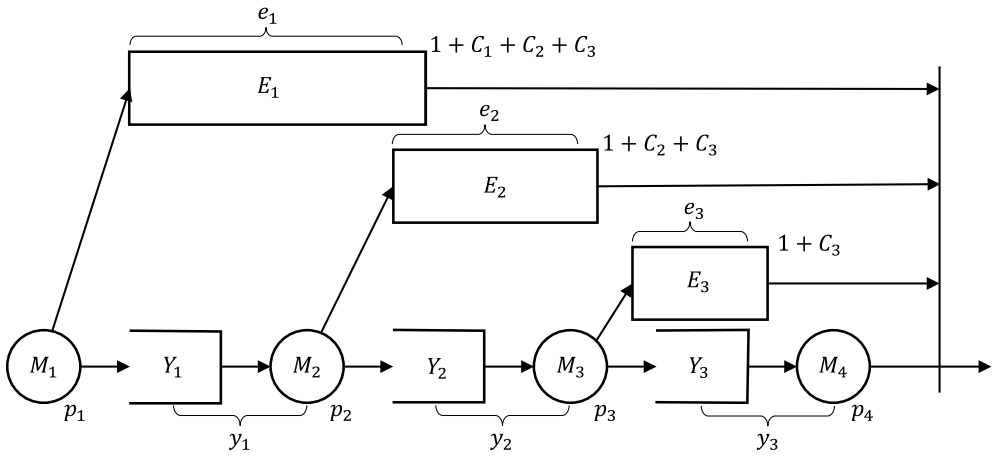

Figure 4. Queueing network model of a serial production line operated under an EB policy.

To highlight the difference between the EB and IB policies, Figure 5 shows the queuing network model of a production line operated under IB that is analogous to the EB model in Figure 4. When a part flows from machine M_n to buffer Y_n, a token is generated and is placed in an associated finite buffer denoted by $I_n, n = 1, \dots, N-1$. The number of tokens in $I_n, n = 1, \dots, N-1$, is denoted by i_n and represents the installation WIP downstream of M_n, i.e., the number of parts that have been produced by machine M_n but have not yet departed from M_{n+1}. Hence, I_n is a surrogate of the installation buffer following M_n (i.e., $B_n \cup M_{n+1}$) in the physical line shown in Figure 1. Note that in the case of the IB policy, the stage WIP levels are identical to the installation WIP levels, i.e., $y_n = i_n, n = 1, \dots, N-1$.

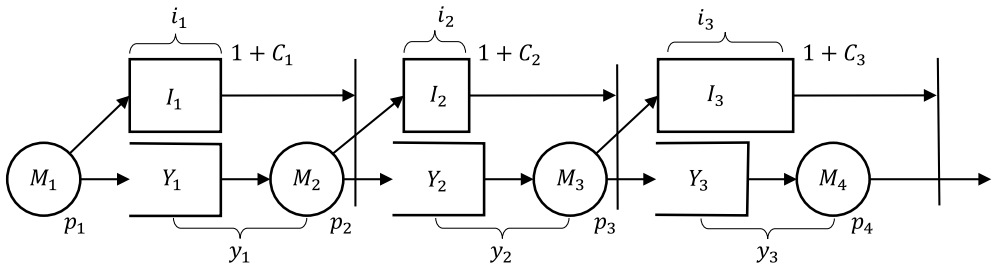

Figure 5. Queueing network model of a serial production line operated under an IB policy.

An important advantage of the EB policy, besides increasing buffer space utilization, is that it uses global information because it enables each machine to process parts based on the echelon WIP level of the entire part of the line downstream of this machine. The decision to allow or block the production of new parts by a machine based on the echelon WIP level should be economically advantageous especially if the WIP holding cost increases significantly as we move down the line, as is the case in high added value production. In contrast, the IB policy uses only local information because it enables each machine to process parts based on the local installation WIP level immediately following this machine.

Under the EB policy, parts are produced earlier by the first and the last machine of the line than they are under the IB policy; hence, the average throughput of the line should be higher under EB than it is under IB, other things equal. On the downside, parts spend more time in the line under the EB policy than they do under the IB policy as a result of the increased congestion induced by the former policy. From Little's law, this implies that the average WIP in the line should be higher under EB than it is under IB. The question that we address in this paper is whether the benefit of the throughput increase under the EB policy outweighs the disadvantage of the WIP increase, also taking into account that less total buffer space may be needed under the EB policy than under the IB policy to achieve the same throughput level.

The remainder of this paper is organized as follows. In Section 2, we briefly review the related literature on the buffer allocation problem. In Section 3, we set up a constrained optimization problem whose objective is to determine the optimal buffer sizes to maximize the average profit of the line subject to a minimum average throughput constraint, under any operating policy. In Section 4, we present numerical results on the optimal performance of a production line under the IB and EB policies, also comparing them against the CONWIP policy. Finally, in Section 5, we draw conclusions.

2 Literature Review

One of the most widely researched problems in flow line optimization is the buffer allocation problem (BAP). BAP deals with allocating storage capacity to intermediate storage buffers to meet a given criterion under given constraints. In a recent survey on this topic, Demir et *al*. (2014) identify three main BAP variants. In one variant, which is often referred to as primal BAP in the literature, the goal is to minimize the total buffer size to achieve a given desired average throughput. In another variant, which is often referred to as dual BAP, the goal is to maximize the average system throughput for a given fixed total buffer size. In the third variant, the goal is to minimize the average system WIP subject to total buffer size and average throughput constraints. A more recent review of the BAP can be found in Schwarz et *al*. (2017).

According to Tempelmeier (2003), who raised important practical considerations in the optimization of flow production systems, planners normally treat average throughput as a datum and therefore usually consider the primal BAP variant. Recently, Shi (2012) and Shi and Gershwin (2009, 2014) considered an extension of the primal BAP whose objective is to maximize the average profit of the line subject to a given minimum average throughput constraint. The average profit is defined as the weighted average throughput of the system minus the sum of the weighted average WIP plus total buffer capacity. In this paper we consider this variant. This constrained problem includes as special cases: 1) the unconstrained problem, when the minimum average throughput is zero, and 2) the primal BAP, when the weights of the average throughput and the total buffer capacity in the objective function are zero. Note that the unconstrained problem has been used for over 30 years (e.g., Kramer and Love 1970; Smith and Daskalaki 1988; Altiok 1997).

Demir et *al*. (2014) and Shi and Gershwin (2014) also categorize the BAP literature based on the search and performance evaluation techniques used. The search techniques include analytical methods (e.g., Enginarlar et *al*. 2005), DP (e.g., Diamantidis and Papadopoulos 2004), heuristics (e.g., Tempelmeier 2003) and meta-heuristics (e.g., Spinellis et *al*. 2000), among others. In this paper, we adopt the gradient search NLP technique that was developed in Shi and Gershwin (2009). Previous works that also use gradient search for optimization and decomposition for performance evaluation include Gershwin and Schor (2000), Levantesi et *al*. (2001), and Helber (2001).

3 Formulation of Buffer Allocation Problem

We consider an optimization problem, similar to that used in Shi and Gershwin (2009) and Shi (2012). The objective is to design the buffer capacities of a production line under any of the three considered policies (IB, EB, and CONWIP) so as to maximize a net profit function subject to a minimum throughput constraint. The profit function is defined as the weighted average throughput of the system minus the sum of the weighted average WIP plus total buffer capacity. The mathematical formulation of the problem is as follows:

$$\max_{C_1,\dots,C_{N-1}} P(C_1,\dots,C_{N-1}) = r\,\nu(C_1,\dots,C_{N-1}) - \left(\sum_{n=1}^{N-1} h_n \bar{y}_n(C_1,\dots,C_{N-1}) + b\sum_{n=1}^{N-1} C_n\right) \tag{1}$$

subject to

$$\nu(C_1,\dots,C_{N-1}) \geq \nu_{min}, \tag{2}$$

$$C_n \geq 0, \quad n = 1,\dots,N-1, \tag{3}$$

where we have used the following notation:

$P(C_1,\dots,C_{N-1})$: average net profit of the line as a function of $C_1,\dots,C_{N-1}$ (\$ per unit time).
$\nu(C_1,\dots,C_{N-1})$: average throughput of the line as a function of $C_1,\dots,C_{N-1}$ (parts per unit time).
$\bar{y}_n(C_1,\dots,C_{N-1})$: average WIP in stage buffer $Y_n, n = 1,\dots,N-1$, as a function of $C_1,\dots,C_{N-1}$ (parts).
r: gross profit coefficient associated with average throughput $\nu(C_1,\dots,C_{N-1})$ (\$ per part).
h_n: inventory holding cost in stage WIP buffer $Y_n, n = 1,\dots,N-1$, (\$ per part per unit time).
b: cost of storage space (\$ per storage slot per unit time).
ν_{min}: minimum required average throughput (parts per unit time).

Given intermediate buffer capacities $C_1, \dots, C_{N-1}$, the performance measures $v(C_1, \dots, C_{N-1})$ and $\bar{y}_n(C_1, \dots, C_{N-1}),\ n = 1, \dots, N-1$, can be estimated using the decomposition-based approximation method developed in Liberopoulos (2017) for the EB policy (as well as the CONWIP policy which is special case of EB). This method is based on decomposing the original system with N machines and $N-1$ echelon buffers into $N-1$ nested segments and approximating each segment with a 2-machine subsystem that can be analyzed in isolation. For the case where the machines have geometrically distributed processing times (Bernoulli reliability model), each subsystem is modelled as a 2D Markov chain that can be solved numerically. The parameters of the 2-machine subsystems are determined by relationships among the flows of parts through the echelon buffers in the original system. These relationships are solved using an iterative algorithm. Liberopoulos (2017) demonstrated that this method is highly accurate and computationally efficient. The performance the IB policy can be estimated using a simple decomposition algorithm similar to that in Li and Meerkov (2009) for the Bernoulli machine case.

Problem (1)-(3) is a constrained optimization problem. To solve it, we use the two-step algorithm proposed in Shi and Gershwin (2009). Briefly, this algorithm is as follows.

In the first step, we solve problem (1)-(3) without taking into account constraint (2). This problem is called the unconstrained problem. To solve this problem we use an iterative gradient technique that works as follows. In each iteration, given the current design $C_1, \dots, C_{N-1}$ and its average net profit $P(C_1, \dots, C_{N-1})$, for each $n = 1, \dots, N-1$, we compute the increase in the average net profit that would result if we raised the value of C_n by one unit to $C_n + 1$. If the increase in the average net profit is negative for all $n = 1, \dots, N-1$, then there are no more gains to make by increasing the buffer capacities; therefore, we stop and keep the current design as the optimal one for the unconstrained problem. Otherwise, we update the current design to a new design in which the capacity of the intermediate buffer that yielded the largest increase in the average net profit is augmented by one and all other capacities remain the same, and we move on to the next iteration. If the optimal solution of the unconstrained problem solved in the first step satisfies constraint (2), then it is also the solution of the constrained problem (1)-(3); hence, we keep it as the final optimal design and stop. Otherwise, we proceed to the second step.

In the second step, we slightly increment the value of the net profit coefficient r and resolve the unconstrained problem with the incremented value of r; essentially, the increment in r is a Lagrange multiplier that is introduced in (1). If the optimal solution of the unconstrained problem with the new value of r satisfies constraint (2), then it is also the solution of the original constrained problem (1)-(3); hence, we keep it as the final optimal design and stop. Otherwise, we slightly increment the value of r again and repeat the process until the solution of the unconstrained problem with the updated value of r satisfies constraint (2). The resulting design is the optimal design for the constrained problem. To evaluate the average net profit that it yields we must use the original value of r in (1).

We used the above algorithm to solve problem (1)-(3) for several instances of two numerical examples under the IB, EB, and CONWIP policies. To choose reasonable values for the input parameters of these instances ($r, b, h_n, n = 1, \dots, N$, and v_{min}) we used the following additional auxiliary parameters:

- c_0: raw-material cost (\$ per raw part).
- I_c: value-added multiplier per production stage (\$ per \$ of value). A value of $I_c = 1.2$ means that there is 20% added value (due to labor and production costs) on each part at each stage (machine).
- c_n: total cumulative cost per part in stage buffer Y_n, $n = 1, \dots, N-1$, (\$ per part).
- c_N: total cumulative cost per finished part exiting the line (\$ per part).
- I_h: interest rate (\$ per \$ invested per unit time).
- I_r: gross profit margin (\$ per \$) defined as follows: $I_r = (s - c_N)/c_N$, where s is the selling price per finished part of the production line (\$ per part). A value of $I_r = 0.1$ means that the selling price of a finished part is 10% higher than the total cumulative cost of the part, c_N.
- I_b: buffer capacity cost multiplier w.r.t. inventory holding cost h_1 (part per storage slot). A value of $I_b = 2$ means that the cost of a slot for storing one part in an intermediate buffer is equal to the cost of holding 2 parts in stage-1 WIP inventory for one period.
- I_v: minimum required production line efficiency. A value of $I_v = 0.8$ means that v_{min} is 80% of the production probability (rate) of the slowest (bottleneck) machine.

Based on the above definitions, the total cost per part in stage n, the profit/cost input parameters in objective function (1), and the minimum required throughput in constraint (2) are computed as follows:

$$c_n = I_c c_{n-1} = I_c^n c_0, n = 1, \dots, N, \tag{4}$$

$$h_n = I_h c_n = I_h I_c^n c_0, n = 1, \dots, N-1, \tag{5}$$

$$r = I_r c_N = I_r I_c^N c_0, \tag{6}$$

$$b = I_b h_1 = I_b I_h c_1 = I_b I_h I_c c_0, \tag{7}$$

$$v_{min} = I_v \min_{n=1,\dots,N} \{p_n\}. \tag{8}$$

In the next section, we present the input data and the results for the two numerical examples that we investigated.

4 Numerical Results on the Optimal IB, EB, and CONWIP Policies

In Example 1, we consider a production line consisting of $N = 4$ machines and 3 intermediate buffers. Each machine has the same production probability, $p_n = 0.6, n = 1, \dots, 4$; hence the line is balanced. For this example, we defined a nominal instance of the design problem with the following parameter values: $c_0 = 100$, $I_c = 1$, $I_h = 0.01$, $I_r = I_b = 0$, and $I_v = 0.78$. Given these values, the parameters in (1) and (2) are computed from (4)-(8) as follows: $[h_1, h_2, h_3] = [1,\ 1,\ 1]$, $r = b = 0$, and $v_{min} = 0.468$. Since $r = b = 0$ and the inventory holding cost rates are the same for all stages, the nominal instance essentially corresponds to the primal BAP mentioned in Section 2, i.e., to the problem of minimizing the average weighted WIP subject to a minimum throughput constraint. Based on the nominal instance, we generated two other instances with different values of I_v. For each instance, we optimized the intermediate buffer capacities $C_n, n = 1,2,3$, for the IB and EB policies. In the case of CONWIP, we kept the capacities of all intermediate buffers except the last one at zero and optimized the capacity of the last buffer $C_{N-1} = C_3$ which we denote by C.

Table 1 shows the input data and optimization results for all instances of Example 1. The nominal instance (0) appears in the first line. The input parameters are shown in the first few columns. For each instance, the value of the parameter that has been varied w.r.t. to the nominal case is shown in bold. The next columns show the optimal values of the intermediate buffer capacities, $C_n^*, n = 1,2,3$ (for CONWIP, only the optimal value of the last buffer C^* is shown), the optimal average throughput v^*, and the maximum net profit P^*, for the three considered policies. The last two columns ("$\%\Delta P^*$") show the percent increase in the optimal profit of the EB policy w.r.t. the IB policy ("E-I") and CONWIP ("E-C"). The average stage WIP levels $\bar{y}_n^*, n = 1,2,3$, corresponding to the optimized policies, are shown in Table 2.

Table 1. Input data and results for Example 1.

	Input parameters					IB policy			CONWIP policy			EB policy			$\%\Delta P^*$	
#	I_c	I_h	I_r	I_b	I_v	$[C_1^*, C_2^*, C_3^*]$	v^*	P^*	C^*	v^*	P^*	$[C_1^*, C_2^*, C_3^*]$	v^*	P^*	E-I	E-C
0	1	0.01	0	0	0.78	[2,3,4]	0.488	-5.141	5	0.473	-4.500	[0,0,5]	0.473	-4.500	12.467	0
1	1	0.01	0	0	**0.85**	[3,5,6]	0.522	-7.176	8	0.516	-6.750	[0,0,8]	0.516	-6.750	5.942	0
2	1	0.01	0	0	**0.92**	[6,9,11]	0.556	-12.387	15	0.554	-12.000	[0,0,15]	0.554	-12.000	3.135	0

Table 2. Additional results for Example 1.

	IB policy			CONWIP policy			EB policy		
#	$\bar{y}_1^*$	$\bar{y}_2^*$	$\bar{y}_3^*$	$\bar{y}_1^*$	$\bar{y}_2^*$	$\bar{y}_3^*$	$\bar{y}_1^*$	$\bar{y}_2^*$	$\bar{y}_3^*$
1	1.639	1.742	1.760	1.500	1.500	1.500	1.500	1.500	1.500
2	2.144	2.544	2.489	2.250	2.250	2.250	2.250	2.250	2.250
3	3.767	4.315	4.305	4.000	4.000	4.000	4.000	4.000	4.000

From the results in Table 1, we observe that in all instances, the average profit is negative under all three policies because the only non-zero term in (1) is the average weighted WIP cost term. As a result, v^* is the smallest throughput that does not violate constraint (2). We also observe that in all instances, the EB policy significantly outperforms the IB policy, confirming our initial intuition that the EB policy, being a global-information policy, should outperform the local-information IB policy. Finally, in all instances, the optimal EB policy is identical to the optimal CONWIP policy. This is expected because the inventory holding cost is the same for all stages. This means that there is no incentive to block parts from moving down the line except in the first machine.

A question that arises naturally is, are there situations where the EB policy significantly outperforms CONWIP? To answer this question, we recall that the CONWIP policy is a special case of the EB policy. Its advantage is that it can achieve the highest average throughput with the smallest total buffer capacity because

none of the machines, except the first one, is ever blocked. One situation where it might make sense to purposely block a machine from processing a part is if the number of parts downstream of this machine is already large and the WIP inventory holding cost downstream of the machine is notably higher than the respective cost upstream of the machine.

With this in mind, we constructed another example in which the WIP inventory holding cost increases significantly with the stages. More specifically, in Example 2, we consider the same 4-machine line as in Example 1 (i.e., one where each machine has the same production probability $p_n = 0.6, n = 1, \dots, 4$) but with different cost parameters. For this example, we defined a nominal instance of the design problem with the following parameter values: $c_0 = 100$, $I_c = 5$, $I_h = 0.001$, $I_r = 0.01$, $I_b = 0.5$, and $I_v = 0.8$. Given these values, the parameters in (1) and (2) are computed from (4)-(8) as follows: $[h_1, h_2, h_3] = [0.5,\ 2.5,\ 12.5]$, $r = 625$, $b = 0.25$, and $v_{min} = 0.48$. Clearly, in this case, the value added cost per stage, and hence the inventory holding cost per stage, increases significantly with the stage. Based on the nominal instance, we generated several other instances, each time varying one of the input parameters, I_c, I_h, I_r, I_b, and I_v. For each instance, we optimized the buffer capacities for the three policies as we did in Example 1. Table 1 shows the input data and the optimization results for the three instances of Example 2.

Table 3. Input data and results for Example 2.

	Input parameters					IB policy			CONWIP policy			EB policy			$\%\Delta P^*$	
#	I_c	I_h	I_r	I_b	I_v	$[C_1^*, C_2^*, C_3^*]$	v^*	P^*	C^*	v^*	P^*	$[C_1^*, C_2^*, C_3^*]$	v^*	P^*	E-I	E-C
0	5	0.001	0.01	0.5	0.8	[6,6,4]	0.535	289.914	11	0.538	286.703	[4,4,4]	0.532	290.400	0.168	1.290
1	**7**	0.001	0.01	0.5	0.8	[9,8,5]	0.549	1184.420	13	0.547	1168.409	[6,6,5]	0.548	1184.631	0.018	1.388
2	**9**	0.001	0.01	0.5	0.8	[11,10,6]	0.557	3345.770	15	0.554	3297.342	[9,7,6]	0.556	3345.155	-0.018	1.450
3	5	**0.005**	0.01	0.5	0.8	[3,3,2]	0.484	175.587	6	0.491	163.861	[3,2,2]	0.482	177.765	1.240	8.485
4	5	**0.01**	0.01	0.5	0.8	[3,3,2]	0.484	48.584	6	0.491	20.739	[3,2,2]	0.482	54.366	11.901	162.141
5	5	0.001	**0.005**	0.5	0.8	[4,4,3]	0.511	126.830	8	0.516	124.361	[2,3,3]	0.508	127.276	0.352	2.344
6	5	0.001	**0.001**	0.5	0.8	[3,3,2]	0.484	4.858	6	0.491	2.074	[3,2,2]	0.482	5.437	11.901	162.141
7	5	0.001	0.01	**1**	0.8	[5,5,4]	0.528	286.088	11	0.538	283.953	[3,4,4]	0.530	287.501	0.494	1.250
8	5	0.001	0.01	**2**	0.8	[5,5,4]	0.528	279.088	10	0.532	279.664	[2,4,4]	0.526	282.220	1.122	0.914
9	5	0.001	0.01	0.5	**0.85**	[6,6,4]	0.535	289.914	11	0.538	286.703	[4,4,4]	0.532	290.400	0.168	1.290
10	5	0.001	0.01	0.5	**0.9**	[7,6,5]	0.541	289.021	12	0.542	285.668	[4,5,5]	0.543	289.384	0.125	1.301
11	5	**0.005**	**0.005**	0.5	0.8	[3,3,2]	0.484	24.292	6	0.491	10.370	[3,2,2]	0.482	27.183	11.901	162.141

Table 4. Additional results for Example 2.

	IB policy			CONWIP policy			EB policy		
#	$\bar{y}_1^*$	$\bar{y}_2^*$	$\bar{y}_3^*$	$\bar{y}_1^*$	$\bar{y}_2^*$	$\bar{y}_3^*$	$\bar{y}_1^*$	$\bar{y}_2^*$	$\bar{y}_3^*$
1	4.267	3.842	2.273	3.000	3.000	3.000	4.248	3.563	2.237
2	6.335	5.136	2.783	3.500	3.500	3.500	6.147	5.163	2.773
3	7.620	6.347	3.271	4.000	4.000	4.000	8.798	6.169	3.260
4	2.442	2.172	1.340	1.750	1.750	1.750	2.978	1.952	1.325
5	2.442	2.172	1.340	1.750	1.750	1.750	2.978	1.952	1.325
6	3.008	2.655	1.766	2.250	2.250	2.250	2.561	2.572	1.733
7	2.442	2.172	1.340	1.750	1.750	1.750	2.978	1.952	1.325
8	3.582	3.146	2.190	3.000	3.000	3.000	3.552	3.435	2.207
9	3.582	3.146	2.190	2.750	2.750	2.750	2.958	3.245	2.162
10	4.267	3.842	2.273	3.000	3.000	3.000	4.248	3.563	2.237
11	4.916	3.738	2.645	3.250	3.250	3.250	4.557	4.297	2.675

From the results in Table 3, we observe that in all instances except instance 2, the EB policy outperforms both the IB and the CONWIP policies. In some instances the dominance of the EB policy is striking. In instance 2, the underperformance of EB with respect to IB is practically negligible (below 0.02%). We also observe that in several instances, under all polices, $v^*$is notably higher than v_{min}, whereas in other instances, it is slightly above v_{min}. The reason for this difference is that in the former instances, the problem parameters are such that the optimization of the unconstrained problem in the first step of the optimization algorithm yields a high value of v^* that satisfies (2) and thus solves the constrained problem too. In the latter instances, the unconstrained problem yields a value of v^* that does not satisfy (2). In this case, the second step of the algorithm iteratively increments r and resolves the unconstrained problem until v^* becomes just feasible.

5 Conclusions

We compared the performance of the optimal EB policy against the performances of the optimal IB and CONWIP policies, where CONWIP is a special case of the EB policy. Our numerical results showed that EB policy generally outperformed the other two policies, the difference in performance being striking in some

cases. More specifically, when $I_c = 1$, meaning that there is no value added at each stage, the optimal EB policy was identical to the optimal CONWIP policy but significantly outperformed the optimal IB policy. When $I_c > 1$ and I_r is not as high relatively to I_h, the optimal EB policy significantly outperformed the optimal CONWIP policy and performed as well as or better than the IB policy. The EB policy performs well primarily because it increases the utilization of the existing buffers, but can still block machines from producing if the number of parts that they have already produced and is still in the system is large. Clearly, more tests need to be performed to further validate these findings. Another worthwhile direction for future research is to explore how the performances of the three policies are affected if the cost of transferring parts to remote buffers is taken into account. In this case, the dominance of the EB policy over the IB policy is expected to drop while the dominance of the EB policy over CONWIP is expected to increase.

Acknowledgements

This research is expected to be supported by the research project 'Productive4.0 - Electronics and ICT as enabler for digital industry and optimized supply chain management covering the entire product lifecycle'; a conditionally approved project by the EU and BMBF (Call: H2020-ECSEL-2016-2-IA-two-stage).

References

Altiok, T. 1997. *Performance Analysis of Manufacturing Systems*, Springer Series in Operations Research, Springer, New York, NY.

Demir, L., S. Tunali, D. T. Eliiyi. 2014. The state of the art on buffer allocation problem: a comprehensive survey. *Journal of Intelligent Manufacturing* **25** (3) 271-392.

Diamantidis, A. C., C. T. Papadopoulos. 2004. A dynamic programming algorithm for the buffer allocation problem in homogeneous asymptotically reliable serial production lines. *Mathematical Problems in Engineering* **2004** (3) 209–223.

Enginarlar, E., J. Li, S. M. Meerkov. 2005. How lean can lean buffers be? *IIE Transactions* **37** (4) 333-342.

Gershwin S. B., G. E. Schor. 2000. Efficient algorithms for buffer space allocation. *Annals of Operations Research* **93** (1) 117-144.

Helber, S. 2001. Cash-flow-oriented buffer allocation in stochastic flow lines. *International Journal of Production Research* **39** (14) 3061-3083.

Kramer, S. A., R. F. Love. 1970. A model for optimizing the buffer inventory storage size in a sequential production system. *AIIE Transactions* **2** (1) 64-69.

Levantesi, R., A. Matta, T. Tolio. 2001. A new algorithm for buffer allocation in production lines. In *Proceedings of the 3rd Aegean International Conference on Design and Analysis of Manufacturing Systems*, May 19-22, Tinos Island, Greece, pp. 279-288.

Li, J., S. M. Meerkov. 2009. *Production Systems Engineering*, Springer, New York, NY.

Liberopoulos, G. 2017. Performance evaluation of a serial production line operated under a shared echelon buffer policy, Working Paper, Department of Mechanical Engineering, University of Thessaly, Greece.

Schwarz, J., S. Weiss, R. Stolletz. 2017. The buffer allocation problem: A joint classification and review of decision problems, solution approaches, and test instances, Working Paper, Chair of Production Management, Manheim Business School, Manheim University, Germany.

Shi, C. 2012. *Efficient Buffer Design Algorithms for Production Line Profit Maximization*. PhD Dissertation, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Shi, C., S. B. Gershwin. 2009. An efficient buffer design algorithm for production line profit maximization. *International Journal of Production Economics* **122** (2) 725-740.

Shi, C., S. B. Gershwin. 2014. A segmentation approach for solving buffer allocation problems in large production systems. *International Journal of Production Research* **54** (20) 6121-6141.

Smith, J. M., S. Daskalaki. 1988. Buffer space allocation in automated assembly lines. *Operations Research* **36** (2) 342-358.

Spearman, M. L., D. L. Woodruff, W. J. Hopp. 1990. CONWIP: a pull alternative to Kanban, *International Journal of Production Research*, 28 (5) 879-894.

Spinellis, D., C. Papadopoulos, J. M. Smith. 2000. Large production line optimization using simulated annealing. *International Journal of Production Research* **38** (3) 509–541.

Tempelmeier, H. 2003. Practical considerations in the optimization of flow production systems. *International Journal of Production Research* **41** (1) 149–170.