

APPROXIMATE BOUNDS FOR THE BUFFER ALLOCATION PROBLEM USING MATHEMATICAL PROGRAMMING REPRESENTATION

A. ALFIERI

A. MATTA

DSPEA
Politecnico di Torino
via Duca degli Abruzzi 24
10129 Torino - Italy
arianna.alfieri@polito.it

Dipartimento di Meccanica
Politecnico di Milano
via La Masa 1
20156 Milano - Italy
andrea.matta@polimi.it

ABSTRACT: *Optimization via simulation consists in applying iteratively two detached models until an optimality condition is reached: a simulation model for predicting the system performance and an optimization model for finding out the optimal solution. Mathematical programming representation has been recently used to describe the behavior of discrete event systems as well as their formal properties. This paper uses this new general concept of simulation–optimization for developing an algorithm to find approximate bounds for the buffer space allocation problem in serial production lines. Numerical results demonstrate that the approximated bounds are quite strong and drastically reduce the computational time to solve the buffer allocation problem at optimality.*

KEYWORDS: *simulation, linear programming, bounds, flow lines, buffer allocation, optimization.*

1 INTRODUCTION

Simulation is one of the most popular techniques to study the behavior of production systems. It is generally used in all situations in which it is not possible to define mathematical expressions for describing the system behavior. Optimization via simulation consists in applying iteratively two detached models until an optimality condition is reached: a simulation model for predicting the system performance and an optimization model for generating and selecting potential optimal solutions.

Downstream the work of Schruben and Chan (Schruben 2000, Chan & Schruben 2008), a recent paper proposes a mathematical programming representation of production systems that can be used for optimization while their performance measures are contemporarily calculated (Matta 2008). Specifically, the author proposes a model for entering into the black box of the simulation–optimization architecture, possibly reducing, in this way, the computational time needed to reach the optimal solution.

The goal of this paper is to adopt this new general concept of simulation–optimization for developing an algorithm to identify strong approximate bounds for the optimal buffer space allocation in serial production lines. In particular, the buffer allocation problem can be represented by an exact MILP (Mixed Integer

Linear Programming) or an approximate LP (Linear Programming)) formulation. The approximate bounds are based on the LP formulation and allow analysts to identify the most interesting subset of feasible solutions in terms of buffer capacity for each stage of the line. Once the bounds have been developed, they can be used to reduce the computational time needed to find the optimal buffer space allocation using the MILP formulation. The MILP and LP formulations combine simulation and optimization. Even if such formulations represent optimization models, their equations constrain the system behavior as in a simulation run and at the same time allows estimating the system performance. Thus, these formulations behave as standard simulation models inside a simulation–optimization algorithm. The MILP and LP formulations have already been introduced in (Matta 2008). However, the contribution of the paper resides in the strong characterization of a new concept of buffer (the *time buffer* concept) and in exploiting the peculiarities of the formulations, in combination with the time buffer concept, to develop an effective bounding procedure.

The plan of the paper is the following. In section 2 we formally define the problem. Section 3 introduces the concept of time buffer and proposes a procedure to develop lower and upper bounds. Numerical results on random generated instances are reported in section

4. Section 5 concludes the paper.

2 BUFFER ALLOCATION IN FLOW LINES

In this section, we define the production systems we are referring to and describe the buffer allocation problem in such systems. Consider a flow line with J single-machine stages, on which N identical items have to be processed. The processing sequence is known in advance, i.e., no scheduling problem needs to be solved. In particular, item i is processed before item $i + 1$ on each machine of the line. The arrival time of each item, denoted by A_i , and the processing times of each item on each machine j , denoted by t_{ij} , are assumed known in advance. Since no scheduling decision is considered, $A_i \leq A_{i+1}$ for all items i . After having been processed by the first machine, parts proceed to the second machine, then to the third and so forth until the last operation is performed at the last machine; finally parts leave the system. Each machine j , excluding the first, has an incoming buffer B_{j-1} to be dimensioned. The capacity of buffer B_j is denoted with C_j . Part i has to wait in buffer B_{j-1} if machine j is busy in processing another part k (with $k < i$). The blocking before service control rule is assumed for machines (Dallery & Gershwin 1992). Machines are perfectly reliable, i.e., no failure possibility is considered (which is the same as assuming failure possibilities with negligible repair times). Transportation times are considered negligible or already included in machining times. Finally, for sake of simplicity, the last machine is never blocked, thus parts completing their processing can always leave the system.

The buffer allocation problem is a very well known problem both in industrial research and practice. This problem entails two fundamental and intertwined decisions: 1) how much buffer space to allow in a production system (which corresponds to the maximum WIP level accepted in the system); 2) how to distribute such buffer space among the stages of the system.

Depending on the goal pursued during the optimization, two different buffer allocation problems are possible. In the literature, these are referred to as the *primal* and the *dual* problems (Gershwin & Schor 2000). In the *primal* problem, the total cost of the allocated buffer capacity is minimized constrained to a minimum value of expected throughput for the line, P^* :

$$\begin{aligned} \min_{C \in \Omega} \quad & \mathbf{a}^T \mathbf{C} \\ \text{s.t.} \quad & E[P] \geq P^* \end{aligned}$$

where P and $E[P]$ are the system throughput and its expected value respectively, Ω is the finite $(K-1)$ -th

dimensional set of buffer capacities, \mathbf{a} is the cost vector containing costs for each buffer capacity and \mathbf{C} is the vector of buffer capacities. Since the average throughput is an increasing function of buffer dimensions, the optimal solution to the primal problem (i.e., that with minimum cost) corresponds to the minimum feasible value of throughput. Oppositely, in the *dual* problem, the average system throughput is maximized constrained to a maximum budget that is available for the buffer allocation:

$$\begin{aligned} \max_{C \in \Omega} \quad & E[P] \\ \text{s.t.} \quad & \mathbf{a}^T \mathbf{C} \leq a^* \end{aligned}$$

where a^* is the available budget. In this case, the optimal solution will correspond to the maximum level of buffer capacity allowed by the available budget. See Gershwin and Schor (Gershwin & Schor 2000) for a complete description of the buffer allocation problem and the related literature.

These two problems are very general and can be specified for each production system. In the remainder of the paper, we address the *primal* buffer allocation problem in a simple flow line, with a single machine per stage. The MILP formulation for solving exactly the primal problem parameterized on a sample path data set has been proposed in (Matta 2008).

3 LP APPROXIMATE FORMULATION

In this section, the concept of *time buffer* is introduced and a bounding procedure based on *time buffers* is developed.

3.1 TIME BUFFER

The exact MILP formulation of the primal problem (Matta 2008) uses the concept of *space buffer*. The space buffer is what is commonly thought of when speaking of buffer capacity, i.e., an available space, between two adjacent machines, where items can wait to be processed.

An alternative way of modeling the possibility for items to wait in a queue, before being processed on a machine, is the use of the *time buffer*. A time buffer can be considered as the time length an item can be started "in advance" on a machine before the successive machine is available for processing. Given two parts a and b to be processed in the sequence $a \rightarrow b$, a time buffer of capacity s_j in front of machine $j + 1$ forces the starting time of job b at machine j (denoted with x_{bj}) and finishing time of job a on machine $j + 1$ (denoted with $y_{a,j+1}$) in the following way:

$$x_{bj} \geq y_{a,j+1} - s_j$$

The above equation simply says that the job b at machine j can start s_j time units before the completion of job a at machine $j+1$. Notice that the time buffer is different from the slack time in PERT graphs, which is defined as the time available between the estimated completion time of the job and its due date. When the time buffer is null, the two adjacent machines are perfectly synchronized. The main difference between *space* and *time* buffer is the fact that a space buffer of capacity m is always able to accommodate m items, independently from the system conditions. On the contrary, the number of parts that, for instance, time buffer B_j can store is not known a priori; in fact, it depends on the number of parts arriving from machine $j - 1$ in a certain time window (equal to the time buffer) just before machine j ends its processing. Indeed, a time buffer of $m * t$ time units is able to accommodate m items only if all items have processing time (on the machine preceding the buffer) equal to t . Since processing times are not deterministic, a time buffer of $m * t$ will be able, in practice, to accommodate a random number of items, depending on the specific processing times of the m parts.

Therefore, substituting space buffers with time buffers in a flow line introduces an approximation whose accuracy depends on the machines behavior. The more the machines are affected by uncertainty, the less the approximation is valid. However, despite this approximation, the solution of the time buffer problem can be used to find reasonable bounds for the space buffer problem. Indeed, the time buffer problem can be easily modeled using linear programming (LP) and solved with very low computational efforts. Note also that the time buffer concept introduced in this section is different from that adopted in manual assembly lines, which is mainly related to dimensioning the speed of the conveyor between two adjacent machines.

3.2 MATHEMATICAL PROGRAMMING FORMULATION

Let s_{jk} be continuous nonnegative variables representing the time buffer value (downstream machine j) between parts i and $i + k$, and F_{ij} be positive variables representing the finishing time of item i at machine j . The time buffer allocation problem can be modeled as:

Approximate Model SimOpt

$$\begin{aligned} \min \quad & \sum_{j=1}^{J-1} a_j \cdot \sum_k s_{jk} \end{aligned} \quad (1)$$

$$\text{s.t.} \quad F_{i1} \geq A_i + t_{i1} \quad i \quad (2)$$

$$F_{i+1,j} - F_{ij} \geq t_{i+1,j} \quad j, i = 1, \dots, N - 1 \quad (3)$$

$$F_{i,j+1} - F_{ij} \geq t_{i,j+1} \quad i, j = 1, \dots, J - 1 \quad (4)$$

$$F_{i+k,j} - F_{i,j+1} \geq t_{i+k,j} - s_{jk} \quad \begin{array}{l} k \\ i = 1, \dots, N - k \\ j = 1, \dots, J - 1 \end{array} \quad (5)$$

$$F_{NJ} - F_{dJ} \leq T^* \quad (6)$$

$$F_{ij} \geq 0 \quad i, j$$

$$s_{jk} \geq 0. \quad j, k$$

If s_{jk} is positive for some k , it means that a buffer B_j , of some capacity, is needed. We model the time buffer s_{jk} in a way that does not depend on the particular job i , i.e. the time buffer between jobs i and $i + k$ is the same for all i .

Equation (1) is the objective function, i.e., the minimization of the overall buffer costs. Constraints (2) simply state that the completion time of each item on the first machine cannot be smaller than its arrival time to the system plus its processing time. Constraints (3) forbid a machine to process two different parts at the same time, while constraints (4) represent the obvious fact that a part cannot be contemporarily processed by two different machines. Constraints (5) prevent a part from leaving a machine if the immediate downstream time buffer is full. Constraints (6) bound from above the time necessary to produce $N - d$ parts to the available time T^* . Parameter d represents the end of the warm-up identifiable with well-known techniques (Law 2007). Parameter T^* , instead, is related to the minimum throughput, since an upper bound on the production time implies an obvious lower bound on the total produced quantity (Hopp & Spearman 2007). Hence, constraints (6) impose that the throughput must be greater than or equal to a minimum value. Parameters A_i and t_{ij} are assumed either known or sampled from some statistical distributions before solving the LP problem (Schruben 2000).

From computational experiments, it has been observed that in the optimal solution (and in any feasible solution), the values of s_{jk} are, for each j , a non-increasing sequence in k . Consider constraints (5). Item i and item $i + k$ are linked by the following constraint:

$$F_{i+k,j} - t_{i+k,j} \geq F_{i,j+1} - s_{jk}.$$

But the completion time of item i on machine $j + 1$, $F_{i,j+1}$, appears also in the constraint linking item i and item $i + k + 1$:

$$F_{i+k+1,j} - t_{i+k+1,j} \geq F_{i,j+1} - s_{j,k+1}.$$

Terms $(F_{i+k,j} - t_{i+k,j})$ and $(F_{i+k+1,j} - t_{i+k+1,j})$ correspond to the starting times, on machine j , of items

$i + k$ and $i + k + 1$, respectively. Since item $i + k$ is processed before item $i + k + 1$ on each machine, the following condition must hold:

$$F_{i+k,j} - t_{i+k,j} \leq F_{i+k+1,j} - t_{i+k+1,j}.$$

If $s_{jk} \leq s_{j,k+1}$, it would allow item $i + k + 1$ to start processing, on machine j , before item $i + k$. Since item $i + k$ must precede item $i + k + 1$, the additional time $s_{j,k+1} - s_{jk}$ can never be used by item $i + k + 1$ to start. Hence, in the optimal solution, it can never happen that $s_{jk} \leq s_{j,k+1}$. Suppose, by contradiction, that exists an optimal solution s_{jk}^* with $s_{jk}^* \leq s_{j,k+1}$. By reducing $s_{j,k+1}$ to a value smaller than s_{jk}^* , the objective function value can be improved (we are minimizing the sum of s_{jk}) without changing any other variables (i.e., the variables F). Therefore, the solution s_{jk}^* was not the optimum. This reasoning can be applied by induction to any pair of items on each machine, thus proving that in the optimal solution $s_{jk} \geq s_{j,k+1}$, for each j and k .

Moreover, since the production sequence is fixed (i.e., part i processed before part $i + k$, for each i), the time buffers s_{jk} are not additive in k because the time interval s_{jk} includes $s_{j,k+1}$.

From the above considerations, for each machine j , the maximum s_{jk} is s_{j1} and it represents the time buffer needed between machine j and machine $j + 1$. This time, divided by an appropriate average processing time which includes elements of both processing times on machine j and on machine $j + 1$, can be read as the average capacity required for (space) buffer B_j . This value is an approximation for the optimal value found by the MILP formulation, and, for the reasons discussed above, the lower the variability in processing time, the closer the approximation to the optimum. Moreover, as it will be better explained in the next session, the approximate value can be either smaller or larger than the optimum space buffer, depending on the values of processing times in the sample path. However, the information obtainable from the LP solution can be used to find (approximate) lower and upper bounds to the integer space buffer variables in the MILP model, thus improving the computational efficiency, as described in section 3.3.

3.3 BOUNDING PROCEDURE

Once the optimal value for the time buffer s_{j1} for stage j has been calculated, (approximate) upper (UB) and lower (LB) bounds for the space buffer B_j can be obtained by considering the average flow rate of the line.

Upper bound

First of all, notice that the value s_{j1} of the time buffer

is heavily influenced by the variability in processing times of items on machine j and $j + 1$. Critical situations, in the sample path, where machine $j + 1$ is busy on a single item with very long processing time, are already taken into account by the value achieved by s_{j1} in the optimal solution. The solutions s_{j1} can be viewed as an approximation of the mean cycle time to cross buffer j . Using the analogy with the Little's law, an approximate UB can be simply computed dividing s_{j1} by the average processing time, \bar{t}_h , of the fastest machine

$$UB_j = \lceil s_{j1} / \min_{h=1}^J \{\bar{t}_h\} \rceil. \quad (7)$$

Notice that an approximation is introduced in the bound calculation. The values s_{j1} should be appropriately depurated from the processing time components. Unfortunately, the exact composition of s_{j1} , and hence the exact time spent by a part in the buffer, is not easy to find since it depends on the overall system data and not only on the two considered machines.

Lower bound

As far as the lower bound is concerned, the starting point is the minimum throughput required, represented by constraint (6). It is well known that in systems with limited buffers, the smaller the buffer, the larger the decrease in throughput. Moreover, the maximum throughput is always smaller than the production rate of the bottleneck that, in our case (flow lines with single machine stages), corresponds to the slowest machine.

An approximate LB can be then computed by considering the desired throughput in the line, which, for feasibility reason, cannot be larger than the bottleneck rate, and applying the same analogy:

$$LB_j = s_{j1} \cdot P^*. \quad (8)$$

Notice that the bounds described above are approximate bounds, since they are based on average processing times, and hence extreme cases, in which smaller or larger values of space buffer would be optimal, could happen. Moreover, as discussed in section 3, the variables s also include time components dependent on the processing time, that are cumbersome to eliminate. This fact leads to possibly larger values for upper and lower bounds.

Consider the upper bound UB_j . In the continuous case, the value so computed is actually an upper bound since, if constraint $s_{jk} \leq s_{j1}^*$ is added to the problem, the solution will provide values of s that are actually smaller than the optimal value found in the LP solution. Moreover, the throughput of the line

will actually be smaller than the higher processing rate, hence $s_{jk} \cdot E[P] \leq UB_j$. This seems to indicate that in the continuous case, the computed upper bound is actually an upper bound. The fact that s_{j1}^* might include processing time components, is not a problem in the upper bound case (it leads just to a bigger upper bound). However, when moving from continuous to discrete case, the value found so far is only an approximate bound since s_d^* can be smaller or larger than s_c^* (where d and c refer to the *discrete* and *continuous* case, respectively), depending on the sample path.

A similar analysis can be done on the lower bound LB_j . In this case, imposing constraints on the s during simulation leads to $s_{jk} \leq s_{j1}^*$, as in the case of upper bound. Moreover, the effective throughput of the line will be smaller or equal to the desired throughput. This means that, in the continuous case, LB_j will hardly be a lower bound. This behavior (i.e., a too large value to be a lower bound) can be even more accentuated by considering that s_{j1}^* might include processing time components. However, since s_d^* can be smaller or larger than s_c^* , there will be cases in which LB_j will effectively behave as a lower bound.

Two considerations are needed at this point. Firstly, even if approximated, the above described bounds will not be too far from the optimal solution, i.e., in cases when they will cut away the optimum from above or below, the optimum will not be excessively far from the *wrong* bound. This is confirmed by our experiments.

Secondly, even if the lower bound cannot be considered really a bound, especially in the continuous case, a natural lower bound exists: the null buffer. Hence, even if one decides not to rely on LB_j values, lower bounds equal to zero can be used without losing too much in terms of computational time efficiency.

However, the computational experiments reported in section 4 confirm that the bounds so computed are quite strong, in the sense that they do not cut away the optimal solution.

4 TEST CASES

The bounding procedure described in the previous section has been applied to some test cases. Several instances of the primal buffer problem in four distinct test cases have been solved with the LP model described in section 3, and then the approximated bounds have been calculated. In all the experiments the vector cost \mathbf{a} has all unit components and, for simplicity, all parts arrive at time zero.

A first simple test case deals with a tandem machine system with an intermediate buffer to decouple the two production stages. Since processing times are

N	LB	UB
5,000	5	8
10,000	5	8
15,000	6	9
20,000	6	9
25,000	6	9
30,000	6	9
50,000	6	9

Table 1: Two-machine case: approximate bounds and global optimum for a specific sample path

assumed to be exponentially distributed, the system can be modeled as a single server with finite capacity queue and closed form equations can be used to solve exactly the primal problem. Mean processing times are equal to 1 and 1.5 time unit for first and second machine respectively, and the mean requested throughput is equal to 0.746927 parts per time unit. The warm-up length in experiments is equal to 2,000 parts. As an example, Table 1 reports the bounds calculated with the proposed procedure using equations (7) and (8) for a specific sample path with different lengths. The optimum value, obtained analytically for this system, is equal to 7 and fits with the provided bounds. It can be noticed that after $N = 15,000$ the bounds are stable and contain the optimum. One hundred problem instances were randomly created and solved first with the approximate procedure and then with the exact MILP formulation. In all the instances the MILP solution is always contained in the interval identified by the calculated upper and lower bounds (see Table 2). The second test case is related to a three-machine flow line with processing times exponentially distributed. The mean processing rates of machines 1, 2 and 3 are equal to 7, 7 and 6 respectively. We experimented with parameter N taking values between 2,500 (a smaller value does not make any practical sense because $d = 2,000$) and 10,000 parts. The expected time available for production has been selected to reach an overall system efficiency equal to 95%, i.e. $T^* = 1043.5$. Ten different problem instances have been randomly generated for each scenario. Table 3 reports the values of the optimal time buffers, the upper and lower bounds, and the optimal space buffer capacities calculated using the exact MILP formulation (Matta 2008). These results refer to the scenario with N equal to 10,000. The optimal MILP solution is always contained in the interval identified by the computed bounds. The computational time to solve the second problem instance without bounds has been about 135 minutes; the restriction of the solution domain given by the bounds has reduced the computational time to approximately 12 minutes. Similar results were obtained in all the other scenarios. Table 4 shows the bounds calculated for a specific sample path for different simu-

lation lengths. The global optimum for this case has been found out by exhaustive research in a compact domain using a performance evaluation method based on Markov chain decomposition (Levantesi, Matta & Tolio 2003). The global optimum value for this case is $\mathbf{C} = [9, 12]$.

The third case is related to a four-machine flow line with processing times uniformly distributed. The domains of the uniform processing times for the four machines are, from the first to the last machine respectively, $[1;2]$, $[2;2.5]$, $[1;2]$, $[1.25;3.25]$. The warm-up length is equal to 5,000 and parameter T^* has been chosen so as to have an overall system efficiency of 98% and N has been set to 10,000. Ten instances of the same problem have been generated and solved. Table 5 reports the values of the computed bounds and the optimal buffer capacities calculated using the exact MILP formulation. In the fourth run, the MILP solution is outside the interval identified by the bounds; however, another feasible solution having the same objective function value can be found out inside that interval. Because of the assumption on uniform distributed processing times it is not possible to use the method in (Levantesi et al. 2003) for fast calculation of system performance, thus the global optimum was not found out.

The fourth case is a flow line with 5 machines separated by intermediate buffers. Machine processing times are exponentially distributed with rate, from the first to the last machine respectively, equal to 5, 7, 2, 7, 5 pieces per time unit. The requested throughput is equal to 1.7 parts per time unit and the warm-up length is equal to 5,000 parts. Also in this case the global optimum $\mathbf{C} = [2, 3, 3, 2]$, obtained using the performance evaluation method in (Levantesi et al. 2003) combined with an exhaustive research in a compact set, is contained in the multidimensional interval identified by the calculated boundaries in the analyzed instances (see the example in Table 6).

It is worthwhile to notice that the numerical results reported in this section, specifically the robustness of the bounds, are also quite representative of the rest of the experiments not reported in this paper.

5 DISCUSSION AN CONCLUSIONS

This work deals with the buffer allocation problem in manufacturing systems. Its main contribution is the proposal of a method to derive approximate bounds for delimiting the feasible region of the optimization problem. The method exploits an approximate LP formulation of the system that relaxes the integer buffer capacity variables into the continuous domain. Indeed, another contribution of the paper is a full comprehension of the modeled time buffers. This knowledge has led to develop bounds that, on the

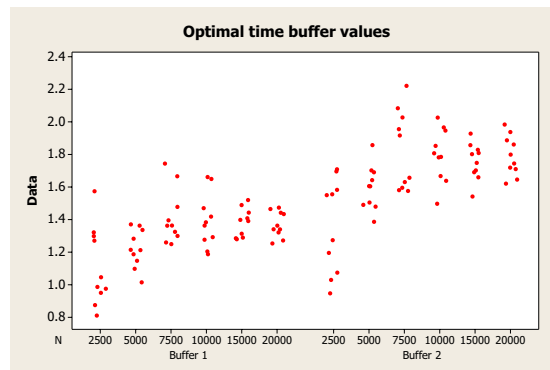


Figure 1: Three-machine case: optimal time buffer vs number of parts.

basis of the results reported in the numerical section, have proved to be strong in the analyzed test cases. Despite the fact that this paper deals specifically with flow lines, we think that the proposed bounding procedure can be extended to other types of manufacturing systems.

Nevertheless many work has to be done yet. First of all the approximation introduced by the continuous model has to be investigated in detail to find some formal ways to increase the confidence of the bounding procedure. Secondly, how to use the bounds in a complete optimization procedure is still not clear. If the procedure uses a unique long run, the analyst should observe how the optimal time buffer allocation evolves as the number of simulated parts increases. These values should converge to a limiting value as in sample-path optimization experiments (Robinson 1996)(Shapiro 1996). A stopping rule can be derived from this analysis. In the other case in which the optimization procedure uses the replication-deletion approach, the optimal time buffers calculated for each independent sample path could be used for building the interval estimate of their mean. This estimate could help to enlarge the bounds thus increasing their confidence.

Figure 1 seems to confirm the practicality of both approaches. The figure shows, for the three-machine case, the graph of the optimal time buffer values as the number of parts increases. From the graph, it seems that the mean values of the optimal time buffers converge, to an asymptotic value, around 15,000 parts and that their variability decreases as the number of parts increases. These considerations will be taken into account for developing a complete optimization procedure by using, in a combined way, the approximate bound procedure to reduce the feasible region and the MILP model to find out the optimum.

References

- Chan, W. & Schruben, L. (2008). Optimization models of discrete-event system dynamics, *Operations Research* **56**(5): 1218–1237.
- Dallery, Y. & Gershwin, S. B. (1992). Manufacturing flow line systems: A review of models and analytical results, *Queueing Systems Theory and Applications, Special Issue on Queueing Models of Manufacturing Systems* **12**(1-2): 3–94.
- Gershwin, S. B. & Schor, J. (2000). Efficient algorithms for buffer space allocation, *Annals of Operational Research* **93**: 117–144.
- Hopp, W. & Spearman, M. (2007). *Factory Physics*, McGraw-Hill.
- Law, A. (2007). *Simulation Modeling and Analysis*, 4th edn, McGraw-Hill.
- Levantesi, R., Matta, A. & Tolio, T. (2003). *Analysis and Modeling of Manufacturing Systems*, Kluwer Academic Publishers, chapter 9 - Performance evaluation of production lines with random processing times, multiple failure modes and finite buffer capacity - Part II: the decomposition, pp. 201–219.
- Matta, A. (2008). Simulation optimization with mathematical programming representation of discrete event systems, in S. J. Mason, R. R. Hill, L. Monch, O. Rose, T. Jefferson & J. W. Fowler (eds), *Proceedings of the 2008 Winter Simulation Conference*, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., pp. 1393–1400.
- Robinson, S. (1996). Analysis of sample-path optimization, *Mathematics of Operations Research* **21**: 513–528.
- Schruben, L. W. (2000). Mathematical programming models of discrete event system dynamics, in J. A. Joines, R. R. Barton, K. Kang & P. A. Fishwick (eds), *Proceedings of the 2000 Winter Simulation Conference*, Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc., pp. 381–385.
- Shapiro, A. (1996). Simulation based optimization—convergence analysis and statistical inference, *Stochastic Models* **12**(3): 425–454.

Run	LB	UB	MILP	Run	LB	UB	MILP
1	5	8	7	51	5	8	8
2	4	7	7	52	6	9	8
3	5	8	7	53	5	8	8
4	5	8	7	54	6	9	8
5	6	9	8	55	5	8	7
6	5	7	7	56	5	8	8
7	6	9	8	57	5	7	7
8	6	9	7	58	5	7	7
9	6	9	7	59	5	8	7
10	7	11	8	60	6	9	8
11	7	10	8	61	4	7	7
12	5	8	8	62	6	9	7
13	6	9	7	63	5	8	7
14	5	8	8	64	5	8	7
15	5	8	7	65	6	9	7
16	6	9	8	66	5	8	7
17	4	7	7	67	7	10	8
18	5	8	7	68	6	9	8
19	4	7	7	69	7	10	8
20	6	8	7	70	5	7	7
21	4	7	7	71	6	9	8
22	6	10	8	72	6	9	8
23	6	9	8	73	4	6	6
24	5	8	7	74	6	9	8
25	6	10	8	75	5	7	7
26	6	9	8	76	6	9	8
27	5	8	8	77	5	7	7
28	5	8	7	78	5	8	7
29	6	9	8	79	5	8	8
30	6	9	8	80	6	9	8
31	6	9	8	81	5	7	7
32	5	7	7	82	6	8	8
33	7	10	8	83	6	10	8
34	6	9	8	84	6	9	8
35	6	8	8	85	5	8	7
36	5	8	7	86	7	10	8
37	6	9	8	87	8	12	8
38	6	9	8	88	6	9	8
39	6	8	8	89	5	8	7
40	6	9	8	90	6	9	7
41	6	9	8	91	7	10	8
42	6	9	8	92	6	8	7
43	6	9	8	93	5	7	6
44	5	8	8	94	9	12	9
45	6	8	8	95	6	9	8
46	8	12	8	96	5	8	7
47	7	10	8	97	7	10	8
48	4	7	7	98	6	9	8
49	6	9	8	99	5	8	7
50	8	11	9	100	6	9	7

Table 2: Case two-machine: approximate bounds and MILP solution for 100 problem instances ($N = 10,000; d = 2000$)

Run	Time buffer		B_1		B_2		MILP	
	B_1	B_2	LB_1	UB_1	LB_2	UB_2	B_1	B_2
1	1.19	1.67	6	9	9	12	8	10
2	1.36	1.78	7	10	10	13	7	12
3	1.66	2.03	9	12	11	15	11	13
4	1.20	1.64	6	9	9	12	8	10
5	1.29	1.81	7	10	10	13	7	12
6	1.28	1.85	7	9	10	13	8	12
7	1.47	1.78	8	11	10	13	8	12
8	1.42	1.97	8	10	11	14	8	13
9	1.65	1.95	9	12	11	14	9	12
10	1.38	1.50	7	10	8	11	8	10

Table 3: Three-machine case: optimal time buffer, bounds and MILP optimal solution ($N=10,000$)

N	B_1		B_2	
	LB_1	UB_1	LB_2	UB_2
5,000	6	9	8	12
10,000	7	10	11	14
15,000	8	11	11	14
20,000	8	11	11	15
25,000	8	11	10	14
30,000	8	11	10	14

Table 4: Three-machine case: approximate bounds for a specific sample path

Run	Time buffer			B_1		B_2		B_3		MILP		
	B_1	B_2	B_3	LB_1	UB_1	LB_1	UB_2	LB_3	UB_3	B_1	B_2	B_3
1	1.99	3.20	3.26	0	2	0	3	0	3	2	2	3
2	1.98	3.26	3.45	0	2	0	3	0	3	2	2	3
3	1.98	2.86	3.16	0	2	0	2	0	3	2	2	3
4	1.99	3.18	3.37	0	2	0	3	0	3	2	4	2
5	1.99	2.73	3.37	0	2	0	2	0	3	2	2	3
6	1.99	2.95	3.28	0	2	0	2	0	3	2	2	3
7	1.99	3.24	3.50	0	2	0	3	0	3	2	2	3
8	1.99	3.16	3.33	0	2	0	3	0	3	2	2	3
9	1.99	3.19	3.41	0	2	0	3	0	3	2	2	3
10	1.98	2.71	3.29	0	2	0	2	0	3	2	2	3

Table 5: Four-machine case: optimal time buffers, bounds and MILP optimal solution ($N=10,000$)

N	B_1		B_2		B_3		B_4	
	LB_1	UB_1	LB_2	UB_2	LB_3	UB_3	LB_4	UB_4
10,000	0	2	0	3	0	3	0	2
15,000	0	2	0	3	0	3	0	2
20,000	0	2	0	3	0	3	0	2

Table 6: Five-machine case: approximate bounds for a specific sample path