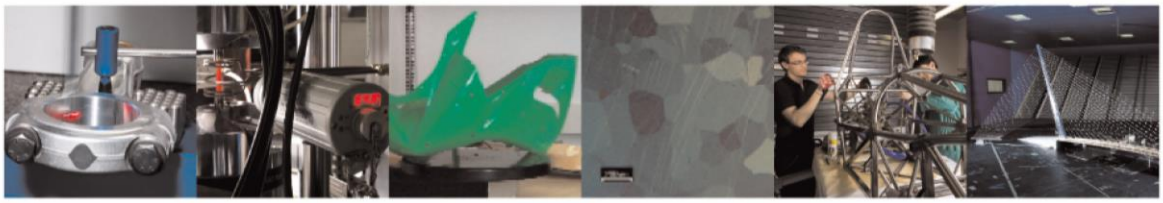


Models and algorithms for throughput improvement problem of serial production lines via downtime reduction

Zhang, M.; Matta, A.

This is an Accepted Manuscript of an article published by Taylor & Francis in IISE TRANSACTIONS on 06 Dec 2019, available online:
https://doi.org/10.1080/24725854.2019.1700431

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

Models and Algorithms for Throughput Improvement Problem of Serial Production Lines via Downtime Reduction

ARTICLE HISTORY

Compiled November 28, 2019

ABSTRACT

Throughput is one of the key performance indicators for manufacturing systems, and its improvement remains an interesting topic in both industrial and academic field. One way to achieve improvement is reducing downtime of unreliable machines. Along this direction, it is natural to pose questions about the optimal allocation of improvement effort to a set of machines and failure modes. This paper develops mixed integer linear programming models to improve system throughput by reducing downtime in the case of multistage serial lines. The models take samples of processing time, uptime and downtime as input, generated from random distributions or collected from real system. To improve computational efficiency while guaranteeing the exact optimality of the solution, algorithms based on Benders Decomposition and discrete event relationships of serial lines are proposed. Numerical cases show that the solution approach can significantly improve efficiency. The proposed modeling and algorithm is applied to throughput improvement of various systems, including a long line and a multi-failure system, and also to the downtime bottleneck detection problem. Comparison with state-of-the-art approaches shows the effectiveness of the approach. Supplementary materials are available for this article. Go to the publisher's online edition of IISE Transaction.

KEYWORDS

Manufacturing systems; serial production line; mathematical programming; discrete event systems; performance improvement

1. Introduction

Throughput is one of the most important performance indicators for manufacturing systems, since it is directly related to company's profitability. Companies usually face with the problem of how to improve the system throughput, which is affected by various factors in complicating way and difficult to analyze. Among the relevant factors, machine unreliability plays an important role. In a case study on a production line of Scania, the increment of throughput was up to

5.5% through repair time reduction (Colledani et al., 2010). Increasing time to failure (uptime) and decreasing time to repair (downtime) are effective ways to achieve throughput improvement through operation and management measures. Compared with high cost of machine upgrading and production loss during system change, operation and management measures are at lower cost and can be conducted more frequently than systems' physical changes. For instance, preventive maintenance can prolong time to failure, whereas developing standard procedures and training operators can be helpful in downtime reduction. This work will focus on improving throughput by downtime reduction in serial lines. Specifically, we will decide the downtime of which failure modes of which machines should be reduced to achieve effective improvement at system level.

The throughput improvement problem is a mixed integer optimization problem. The real-valued aspect is to decide how much each failure should be improved. Considering that the cost may be not linear on the downtime reduction level, 0-1 variables can be introduced for modeling reason. In literature, throughput improvement problems are usually solved by greedy algorithms, including Continuous Improvement (CI) and Gradient Methods (GM). CI (Kang, Zhao, Li, & Horst, 2016; J. Li, 2013) is a commonly applied approach, and the procedure is repeatedly identifying and mitigating the bottleneck machine in the system. The bottleneck machine is defined as the machine by improving whose performance can result in the largest increase of system throughput (J. Li & Meerkov, 2008). Since the performance of an unreliable machine is characterized by three independent variables, i.e., cycle time, downtime and uptime, bottleneck identification methods may find different types of bottleneck accordingly (J. Li & Meerkov, 2008). To improve throughput by reducing downtime using CI, the Downtime Bottleneck (DT-BN) machine is identified first. Many works of DT-BN identification can be found in literature, for instance, the arrow method (Chiang, Kuo, & Meerkov, 2000) and the turning point method (Li, 2009). Both methods are developed based on the observation that the bottleneck machine causes the blockage of upstream machines and the starvation of downstream machines. They use estimation of machine blockage and starvation probability to analyze the source of blockage and starvation, then bottleneck indicators are calculated from a systematic point of view. After identifying the DT-BN machine, downtime of the bottleneck is reduced by a certain amount. The identification and mitigation of the DT-BN can be repeated for several iterations. CI is simple, general and convenient to apply, but its shortages are not negligible. First, to the knowledge of the authors, all bottleneck detection methods for multi-stage production systems are heuristic methods, and in some cases the bottleneck cannot be correctly detected (Chiang et al., 2000; Yu & Matta, 2016). Furthermore, the failure level DT-BN detection approach for systems with

multiple failure modes does not exist. Second, CI is myopic, since an approach to determining the appropriate amount of downtime reduction at each iteration has not been addressed. This may lead to insignificant improvement and mitigation effort waste. Third, CI lacks of modeling structure and has limited capability to answer questions such as what a plant manager can do to achieve 10% of system throughput improvement with the lowest cost.

GM estimates the gradient of the system throughput over the downtime of each failure mode of each machine, and moves along the gradient direction with a certain step length. Since serial production lines are complex stochastic systems, and there is no closed form representation for throughput evaluation, deriving stochastic gradient estimator is challenging (Fu et al., 2015). Infinitesimal Perturbation Analysis (IPA) is an efficient approach for gradient estimation (Ho & Cao, 2012). Using IPA, all the elements in the gradient vector can be derived from only one simulation sample path. Using IPA estimator combined with GM, Ho and Cao (1983) addressed the problem of maximizing throughput of serial lines by allocating mean service time to each machine. However, IPA only accelerates the gradient estimation procedure and GM still requires extensive number of iterations for finding the optimum. Moreover, GM is only suitable for continuous functions, but not MILP.

This work contributes to literature in several ways. One is the definition of formal models. More specifically, two problems are addressed, which are how to minimize the cost to achieve a target throughput, and how to achieve the maximal throughput within a downtime reduction budget. Discrete Event Optimization (DEO) (Pedrielli, Matta, Alfieri, and Zhang (2018)), an integrated simulation-optimization framework, is used to model and solve the above mentioned problems. DEO has been successfully applied to many production system optimization problems (Matta (2008); Pedrielli, Matta, and Alfieri (2015); Tan (2015); Weiss and Stolletz (2015); Zhang, Matta, Alfieri, and Pedrielli (2017); Zhang, Matta, and Pedrielli (2016)). Under the DEO framework, throughput evaluation via discrete event dynamics and an optimization problem can be included in a unique Mixed Integer Linear Programming (MILP) model. To deal with the computation issue of MILP, this work proposes algorithms based on Benders Decomposition (BD) and discrete event relationships of production systems. The developed approach saves the computational time, and solution optimality is also guaranteed, which represents the second contribution of this work. Another original contribution is that a model of downtime bottleneck detection problem at failure mode level is formulated as a variation of the throughput improvement model, and the proposed solution approach is still effective.

The remainder of this paper is organized as follows. The problems are defined and modeled

in section 2 and section 3. The solution approach is described in section 4. The downtime bottleneck detection problem is presented in section 5. The numerical analysis is applied to several cases, and reported in section 6. Finally, the conclusion is addressed in section 7.

2. Problem definition

2.1. Serial production line with unreliable machines with multiple failure modes

The production system analyzed in this work is the serial line as shown in Figure 1, composed by M unreliable machines and inter-machine buffers of finite capacity. All parts will be processed by all the machines along the line, following the arrival sequence. There are always parts available in front of the first machine, and parts can always leave the system after being processed by the last machine.

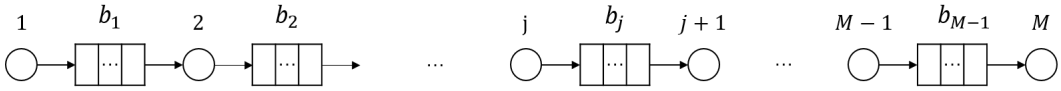


Figure 1.: Serial line with M machines and $M-1$ finite buffers.

The inter-machine buffers have finite capacities b_j, with a vector notation $\mathbf{b} = [b_1, ..., b_{M-1}]^T$. A full buffer causes the *blockage* of the upstream machine. An empty buffer causes the *starvation* of the downstream machine. We consider the production system with *block-after-service* behavior, which means that machine j may release a part to the downstream buffer j only if there is at least one space available.

Failures may occur on unreliable machines. Unreliable machines have two types of state: up and down. Machines produce parts in the up state with a specific cycle time, denoted by vector $\mathbf{p}$, and stop working in the down state because of failure. Failure is characterized by uptime and downtime. Uptime refers to the period between the repair completion and the next failure, denoted by vector $\mathbf{T_{up}}$. Downtime refers to the time for repairing a failed machine, denoted by vector $\mathbf{T_{down}}$. We assume that all the failures are operation-dependent, i.e., the uptime will take into account the processing period only, and blocking and starvation periods will be excluded. Multiple failure modes may exist at the same machine. For example, stuck of component feeding equipment and tool failure have their own uptimes and downtimes. In the remainder of this paper, we use the term "failure" to refer to a failure mode of a machine. We assume cycle time, uptime and downtime are all random variate, and do not specify a particular distribution. Since sample-based method is applied, it is important to distinguish *distribution*

and *samples* by different terms for sake of clarity. Specifically, we use cycle time, uptime and downtime to refer to the *distributions*, and processing time, time to failure and repair time to refer to the *samples*.

2.2. Throughput improvement problems

Throughput of serial lines with unreliable machines is a function of buffer capacity $\mathbf{b}$, cycle time $\mathbf{p}$, uptime $\mathbf{T_{up}}$ and downtime $\mathbf{T_{down}}$, denoted by $TH(\mathbf{b}, \mathbf{p}, \mathbf{T_{up}}, \mathbf{T_{down}})$. To improve throughput, one can reallocate buffer spaces, reduce cycle time by upgrading machines, increase uptime by implementing preventive maintenance, or reduce downtime by improving maintenance activity. This work only addresses throughput improvement by reducing downtime.

Implementing downtime reduction measures will need cost in general sense, measured in terms of financial expense, human resource or time according to real situations. The cost is related to the reduction amount and denoted by $\mathcal{B}(\mathbf{T}^{'}_{\mathbf{down}})$, where $\mathbf{T}^{'}_{\mathbf{down}}$ is the reduced new downtime. It is difficult to achieve the maximal throughput improvement and minimal total cost at the same time, so two optimization problems can be defined, which will be the identification of failures whose downtime should be reduce and how much is the reduction amount, to achieve the maximal throughput within budget (B^*) or to minimize the cost achieving target throughput (TH^*), as formulated in (1) and (2). The two problems are named throughput maximization problem denoted by MP1 and cost minimization problem denoted by MP2, respectively.

$$\max\{TH(\mathbf{b}, \mathbf{p}, \mathbf{T_{up}}, \mathbf{T}^{'}_{\mathbf{down}}) | \mathcal{B}(\mathbf{T}^{'}_{\mathbf{down}}) \leq B^*\} \tag{1}$$

$$\min\{\mathcal{B}(\mathbf{T}^{'}_{\mathbf{down}}) | TH(\mathbf{b}, \mathbf{p}, \mathbf{T_{up}}, \mathbf{T}^{'}_{\mathbf{down}}) \geq TH^*\} \tag{2}$$

3. Mathematical models

The DEO models for throughput improvement problems, i.e., MP1 and MP2, are presented in this section, together with a note on modeling and application issues for practitioners. The parameters used in the mathematical models are as follows:

Parameters

M	Machine number.
j	Machine indices, $j = 1, ..., M$.
N	Total number of parts.
i	Part indices, $i = 1, ..., N$.
$\mathcal{F}_j$	The set of failures of machine j.
k	Failure indices, $k = 1, ..., \mathcal{F}_j$.

Parameters (continue)

q	Index for failure occurrences, when several failures of the same mode occur when a machine processes a single part.
$r_{i,j,k,q}$	Repair time of the q-th failure occurrence of type k that part i encounters at machine j, if $r_{i,j,k,q} > 0$; otherwise if the part encounters fewer than q failures, $r_{i,j,k,q} = 0$.
$\mathcal{Q}$	The set of (i, j, k, q) such that $r_{i,j,k,q} > 0$.
b_j	Buffer space between machine j and $j + 1$.
$p_{i,j}$	Processing time of part i at machine j.
B^*	Downtime reduction budget.
TH_0	Original throughput.
ΔTH^*	Target system throughput improvement ratio, and target throughput is equal to $TH_0(1 + \Delta TH^*)$.
$U_{j,k}$	Upper bound of downtime reduction of failure mode k of machine j.

In DEO models, variables of event occurring times and optimization variables are both essential. Variables $e^s_{i,j}$ and $e^d_{i,j}$ represent the occurring time of starting events and departure events, and variables $x_{j,k}$ reflect the optimization aspect, which is how much the downtime is reduced. Variables, such as $t_{i,j},\ r^{'}_{i,j,k,q},\ z,\ B$, are actually redundant and introduced for model clarity reason, and they can be directly calculated given $x_{j,k}$.

Variables

$0 \le x_{j,k} \le U_{j,k}$	Coefficient of downtime reduction amount at machine j of failure mode k.
$\mathbf{x}$	Vector representation of $x_{j,k}$.
$t_{i,j} \ge 0$	The total delay time of part i at machine j, which is equal to the sum of the processing time and the repair time.
$r^{'}_{i,j,k,q} \ge 0$	Repair time after reduction of the q-th failure of type k that part i encounters at machine j.
z	Reciprocal of system throughput.
B	Total cost of the downtime reduction.
$e^s_{i,j} \ge 0$	Starting time of part i at machine j.
$e^d_{i,j} \ge 0$	Departure time of part i at machine j.

The DEO model of throughput maximization problem (MP1) of serial production lines is shown below. The model is denoted by MP1-O.

$$min\{z\}$$

$$s.t.$$

$$z = \frac{e^d_{N,M}}{N} \tag{3}$$

$$e^d_{i,j} - e^s_{i,j} \ge t_{i,j} \qquad \forall i = 1, ..., N,\ j = 1, ..., M \tag{4}$$

$$e^s_{i,j} - e^d_{i-1,j} \ge 0 \qquad \forall i = 1, ..., N,\ j = 1, ..., M \tag{5}$$

$$e^s_{i,j} - e^d_{i,j-1} \ge 0 \qquad \forall i = 1, ..., N,\ j = 1, ..., M \tag{6}$$

$$e^d_{i,j} - e^s_{i-b_j,j+1} \ge 0 \qquad \forall i = 1, ..., N,\ j = 1, ..., M \tag{7}$$

$$t_{i,j} = p_{i,j} + \sum_{k,q:\ (i,j,k,q)\in\mathcal{Q}} r'_{i,j,k,q} \qquad \forall i = 1,...,N,\ j = 1,...,M \tag{8}$$

$$r'_{i,j,k,q} = f_{j,k}(x_{j,k}, r_{i,j,k,q}) \qquad \forall (i,j,k,q) \in \mathcal{Q} \tag{9}$$

$$B \leq B^* \tag{10}$$

$$B = \mathcal{B}(\mathbf{x}) \tag{11}$$

$$0 \leq x_{j,k} \leq U_{j,k} \qquad \forall j = 1,...,M,\ k \in \mathcal{F}_j \tag{12}$$

$$e^s_{i,j},\ e^d_{i,j},\ t_{i,j} \geq 0 \qquad \forall i = 1,...,N,\ j = 1,...,M \tag{13}$$

$$r'_{i,j,k,q} \geq 0 \qquad \forall (i,j,k,q) \in \mathcal{Q} \tag{14}$$

The objective function is the minimization of z, the reciprocal of throughput, which is equivalent to throughput maximization. Constraint (3) indicates system throughput equal to the number of parts produced per unit time calculated between time 0 and $e^d_{N,M}$. Constraints (4) to (7) represent the event relationships in the simulation model of a serial production line. Each object in the system, i.e., a part, a machine or a buffer, goes through a specific discrete event flow, as in Figure 2. In the event flow of part i and machine j, there is a delay time $t_{i,j}$, that the part spends in the machine, as in constraints (4). A machine processes the parts sequentially, as in constraints (5). A part enters and leaves the machines sequentially, as in constraints (6). Machine starvation can be observed if $e^s_{i+1,j} > e^d_{i,j}$, and blockage can be observed if $e^d_{i,j} > e^s_{i,j} + t_{i,j}$. A buffer j goes through an event flow composed by departure events from upstream machine j and starting events of downstream machine $j+1$. The buffer level should be between 0 and its capacity b_j. To avoid negative buffer level, as in the event flow of buffer $(j-1)$ and constraints (6), the start of part $(i+1)$ at the downstream machine j should be not earlier than it departs from the upstream machine $(j-1)$. To keep buffer level within its capacity, as in the event flow of buffer j and constraints (7), the departure of part i from the upstream machine j is enabled only after the starting event of part $(i-b_j)$ at machine $j+1$, namely there is a space available. Technically, the footnotes of $e^s_{i,j}$ and $e^d_{i,j}$ are defined as $j \in \{1,...,M\}$ and $i \in \{1,...,N\}$, and in cases $i \notin \{1,...,N\}$ or $j \notin \{1,...,M\}$, the values of $e^s_{i,j}$ and $e^d_{i,j}$ are set to 0.

Constraints (8) calculate the delay time $t_{i,j}$ of part i at each machine j as sum of processing time and repair time. Constraints (9) define *the downtime reduction function* $f_{j,k}$. The samples $r_{i,j,k,q}$ from the same machine j and failure k are reduced via the same function $f_{j,k}$ with coefficient $x_{j,k}$. (10) shows that the cost should be within the budget B^*, and (11) defines the

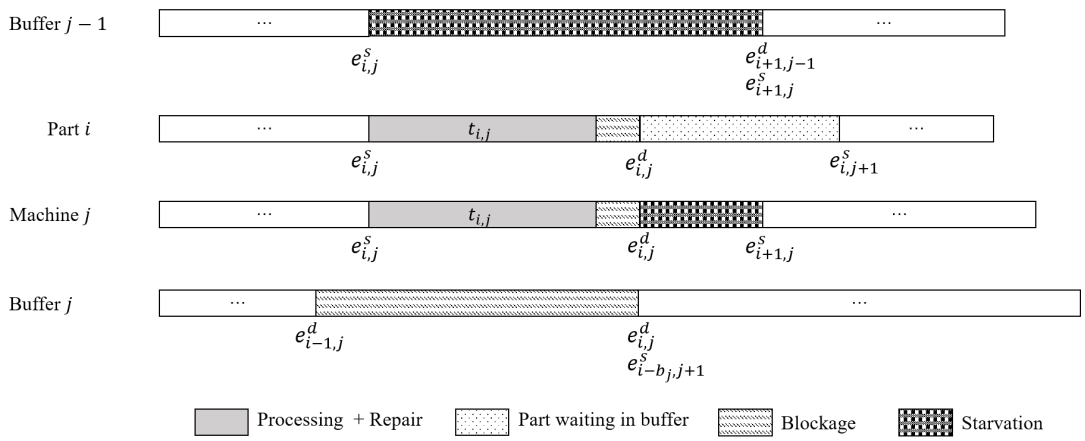


Figure 2.: Event flow of serial lines.

total cost of the downtime reduction using *a cost function* $\mathcal{B}(\mathbf{x})$. The details of modeling the downtime reduction function and the cost function based on real world settings are stated in section 3.1.

The cost minimization problem is modeled as follows:

$$\begin{aligned} & min\{B\} \\ & s.t. \\ z \leq & \frac{1}{TH_0(1+\Delta TH^*)} \end{aligned} \tag{15}$$

$$(3),\ (4),\ (5),\ (6),\ (7),\ (8),\ (9),\ (11),\ (12),\ (13),\ (14),$$

where the objective is to minimize the total cost, and that (15) indicates the resulting system guaranteeing the target throughput. The model is denoted by MP2-O.

In both MP1-O and MP2-O, a transient period appears at the beginning of the simiulation, but the initial bias will be mitigated, as the simulation length increases (Robinson, 1996).

3.1. A note for practitioners

Formulate the models, MP1-O and MP2-O, involves the following steps. First, the distributions of cycle times, uptimes and downtimes should be obtained. They can be theoretical or empirical distributions fitted from the real data. Second, the samples of processing time $p_{i,j}$ and repair time $r_{i,j,k,q}$ are generated from the distributions, the system is simulated with original repair times, and original throughput is fixed as a parameter. Finally, the downtime reduction function (9) and the cost function (11) are defined. The details on modeling this two functions are shown

in section 3.1.1 and 3.1.2, respectively.

3.1.1. Modeling the downtime reduction function

In function (9), the downtime reduction is realized through a coefficient $x_{j,k}$, depicting how the Cumulative Distribution Function (CDF) is changed. $x_{j,k}$ can be used in three alternative ways as shift, scale or combination parameters.

With a **shift** parameter, the downtime reduction is defined as

$$f_{j,k}(x_{j,k}, r_{i,j,k,q}) = r_{i,j,k,q} - x_{j,k} \quad \forall (i,j,k,q) \in \mathcal{Q}. \tag{16}$$

The CDF is shifted to left by $x_{j,k}$, i.e., $r_{i,j,k,q}$ are reduced by the same amount, so that the mean of downtime is reduced but the variance remains unchanged. For instance, downtime from uniform distribution on interval $(2, 4)$ is reduced to uniform distribution on interval $(1, 3)$ by a shift $x_{j,k} = 1$. In this case, $U_{j,k}$, the upper bound of $x_{j,k}$, cannot be greater than the lower limit of the distribution. In practice, this type of reduction can be realized by simplifying some steps of the maintenance procedure, for instance, moving the device for maintenance to a more convenient location or having spare parts available in warehouse.

With a **scale** parameter, the downtime reduction function is defined as

$$f_{j,k}(x_{j,k}, r_{i,j,k,q}) = (r_{i,j,k,q} - a)(1 - x_{j,k}) + a \quad \forall (i,j,k,q) \in \mathcal{Q}, \tag{17}$$

where a is the lower limit of the distribution. The CDF is scaled by a ratio $(1 - x_{j,k})$, i.e., both mean and variance are decreased, but the lower limit is unchanged. For instance, downtime from uniform distribution on interval $(2, 4)$ is reduced to uniform distribution on interval $(2, 3.5)$ by a scale 0.75 with $x_{j,k} = 0.25$. In this case, $U_{j,k}$ cannot be greater than one. In practice, this type of reduction can be realized by analyzing how the maintenance is done in the best case and setting it as the standard operation procedure to reduce the uncertainty.

With a **combination** parameter, the downtime function is defined as

$$f_{j,k}(x_{j,k}, r_{i,j,k,q}) = (r_{i,j,k,q} - a)(1 - x_{j,k}) + a - a_0 x_{j,k} \quad \forall (i,j,k,q) \in \mathcal{Q}, \tag{18}$$

where a is the lower limit of the distribution and a_0 is a shift parameter. The CDF is first scaled by a ratio $(1 - x_{j,k})$ keeping the lower limit unchanged, and then it is shifted to left by

$a_0 x_{j,k}$. Obviously, it is a combination of scale and shift transformation. In this case, $U_{j,k}$ can be neither greater than one nor greater than a/a_0. If $a_0 = a$, (18) will be as $f_{j,k}(x_{j,k}, r_{i,j,k,q}) = r_{i,j,k,q}(1 - x_{j,k})$, and $x_{j,k}$ can be interpreted as the percentage the downtime should be reduced. In this case, the coefficient of variation does not change during the downtime reduction.

In (16), (17) and (18), the larger the value of $x_{j,k}$, the more downtime is reduced, even though $x_{j,k}$ does not represent the downtime reduction itself in the scale or combination cases.

3.1.2. Modeling the cost function

Similarly, the cost function have different structures according to the actual situation. Specifically, cost B is considered as discrete or piecewise linear function over downtime reduction coefficient $\mathbf{x}$. 0-1 or real-valued auxiliary variables may be introduced for model linearization.

In a **discrete** cost function, $x_{j,k}$ is defined on a finite set of discrete values. It refers to the situation in which some alternative downtime reduction proposals have been defined by the manager, but only a subset can be selected for budget or cost saving reason.

In a **piecewise linear** cost function, $x_{j,k}$ is defined continuously on the interval $[0, U_{j,k}]$, and it can be used as an approximation of general functions. For instance, when a repair activity involves an outsourcing supplier, the cost is positively related to the lead time which can be assumed as continuous variable. It refers to the situation in which the manager does not have specific proposals, but he/she would like to know which failures should be analyzed and which level of reduction of the selected failures should be achieved.

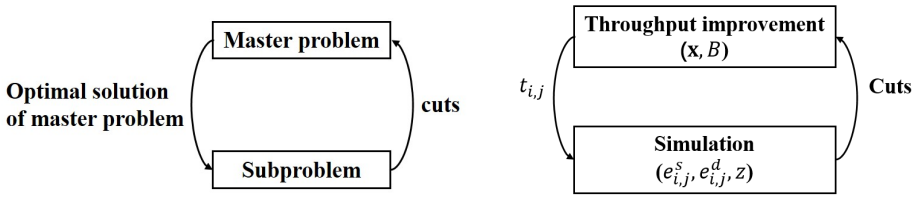
Practically, **discontinuity** at $x_{j,k} = 0$ may appear. If $x_{j,k} = 0$, nothing should be done with this failure, and cost will be equal to zero. Otherwise, the cost for any positive $x_{j,k}$ may not be infinitely close to zero, even though $x_{j,k}$ may be infinitely close to zero in the case of piecewise linear function. Thus, it is natural to introduce a fixed cost in case $x_{j,k} > 0$, and it can interpreted as a manager would not like to involve more failures if the same throughput can be achieved by improving some already selected failures within a reasonable tolerance. For instance, if improving one failure mode by $x = 0.15$ provides the same throughput as improving two failures by 0.1 and 0.01 respectively, the manager may prefer the first plan. Fixed cost equal to 10 and the slope for $x > 0$ equal to 100 can make the first plan preferable.

The formulation of discrete cost function and piecewise linear function with discontinuity at $x_{j,k} = 0$ can be found in Appendix I in the online supplementary material.

4. Solution approach: simulation based Benders cut generation

The scale of the DEO models, MP1-O and MP2-O, increases as the number of parts N and the number of machines M increase. Specifically, there are more than $2MN$ variables and more than $5MN$ constraints. As a result, in computation aspect, it is difficult to solve directly MP1-O and MP2-O of long serial lines or for long simulation. If 0-1 auxiliary variables are used in the cost function, the models become MILP, which is even much harder to tackle.

This computation challenge in DEO has drawn great attention in literature. Weiss and Stolletz (2015) solved the sample path based buffer allocation problem of serial lines exactly and efficiently by applying the Benders Decomposition (BD) (Benders (1962)), a mathematical model partitioning technique, to the DEO model. The idea of BD is to project the feasible set of the original model onto a space of lower dimensions, and to solve the resulting low dimension model. Projecting the entire feasible set brings high computational burden, so the hyperplanes are iteratively projected via generated cuts. The set of variables of the original model are partitioned into two subsets, namely variables of the master problem and of the subproblem. The master problem space is the low dimension space where the original feasible set is projected. The master problem is firstly formulated by removing all the constraints containing subproblem variables, and cuts generated from subproblem are iteratively added to it. The subproblem is formulated by fixing values of master variables, which in standard BD procedure are the optimal solution of the master problem, as in Figure 3(a). The subproblem should be LP, while there are no specific requirements for the master problem. The iterative procedure stops until the optimum of the original problem is found.



(a) Benders decomposition framework.

(b) Model decomposition of the throughput improvement problems.

Figure 3.: Model decomposition framework.

The DEO models in Section 3 will be decomposed based on BD framework in section 4.1, and a simulation-based Benders cut generation algorithm is developed in section 4.2, which further improves the computation efficiency. In section 4.3, the overall solution approach is summarized.

4.1. Model decomposition

The DEO models of throughput improvement problems shown in the previous section are characterized as fully integrating simulation with optimization in a unique model, which contains huge number of real-valued simulation variables $e^s_{i,j},\ e^d_{i,j}$ and much smaller number optimization variables $\mathbf{x}$, and possibly some auxiliary variables introduced in the cost function. Once the optimization variables are fixed, the remaining model is an LP with special structure, namely the event trajectory of a simulation run. If the original DEO model is partitioned as shown in Figure 3(b), i.e., $x_{j,k}$ and B are master problem variables (auxiliary variables are implicitly included in B), whereas $e^s_{i,j}$, $e^d_{i,j}$ and z are subproblem variables, the master problem will contain only a few variables and the subproblem can be handled by simulation other than state-of-the-art LP solvers. Thus, the computational complexity may be decreased. Since $t_{i,j}$ and $r'_{i,j,k,q}$ can be calculated using equations (8) and (9), they are not involved in the decomposition framework.

For MP1, since the variable in the objective function of the original problem, namely z, is included only in the subproblem, the feedback given by the subproblem will always be an optimality cut. The subproblem denoted by MP1-S is formulated as follows:

$$
\begin{aligned}
& min\{z\} && \\
& s.t. && \\
& z = \frac{e^d_{N,M}}{N} && : \vartheta && (19)\\
& e^d_{i,j} - e^s_{i,j} \geq t_{i,j} && : u_{i,j} \quad \forall i = 1,...,N,\ j = 1,...,M && (20)\\
& e^s_{i,j} - e^d_{i-1,j} \geq 0 && : v_{i,j} \quad \forall i = 1,...,N,\ j = 1,...,M && (21)\\
& e^s_{i,j} - e^d_{i,j-1} \geq 0 && : s_{i,j} \quad \forall i = 1,...,N,\ j = 1,...,M && (22)\\
& e^d_{i,j} - e^s_{i-b_j,j+1} \geq 0 && : w_{i,j} \quad \forall i = 1,...,N,\ j = 1,...,M && (23)\\
& e^{\xi}_{i,j} \geq 0 && : \alpha^{\xi}_{i,j} \quad \forall \xi \in \{s,d\} i = 1,...,N,\ j = 1,...,M && (24)
\end{aligned}
$$

where $u_{i,j},\ v_{i,j},\ s_{i,j},\ w_{i,j},\ \vartheta,\ \alpha^{\xi}_{i,j}$ are dual variables. The master problem is as follows:

$$
\begin{aligned}
& min\{\psi\} \\
& s.t. \\
& \mathcal{B}(\mathbf{x}) \leq B^* \\
& \psi \geq \sum_{j=1}^{M} \sum_{k \in \mathcal{F}_j} \sum_{k,q:\ (i,j,k,q) \in \mathcal{Q}} f_{j,k}(x_{j,k}, r_{i,j,k,q}) \bar{u}^l_{i,j} + \sum_{i=1}^{N} \sum_{j=1}^{M} p_{i,j} \bar{u}^l_{i,j} \quad \forall\ l, && (25)
\end{aligned}
$$

with optimality cuts (25) generated from each iteration l, and ψ representing the objective function in the master problem.

For the cost minimization problem, since the variable in the objective function of the original problem, namely B, is included only in the master problem, the feedback given by the subproblem will always be a feasibility cut. The subproblem, more specifically the feasibility subproblem, denoted by MP2-S is formulated as follows:

$$\begin{aligned} & min\{\varepsilon\} \\ & s.t. \\ z - \varepsilon \leq \frac{1}{TH_0(1+\Delta TH^*)} \quad & : \ \vartheta \\ \varepsilon \geq 0 & \end{aligned}$$

$$(19),\ (20),\ (21),\ (22),\ (23),\ (24)$$

The master problem is:

$$\begin{aligned} & min\{B\} \\ & s.t. \\ & B = \mathcal{B}(\mathbf{x}) \\ 0 \geq \sum_{j=1}^{M} \sum_{k \in \mathcal{F}_j} \sum_{k,q:\ (i,j,k,q) \in \mathcal{Q}} & f_{j,k}(x_{j,k}, r_{i,j,k,q}) \bar{u}_{i,j}^{l} + \sum_{i=1}^{N} \sum_{j=1}^{M} p_{i,j} \bar{u}_{i,j}^{l} - \frac{1}{TH_0(1+\Delta TH^*)} \quad \forall\ l,\ (26) \end{aligned}$$

with feasibility cuts (26) generated from each iteration l.

4.2. Simulation based cut generation

To formalize Benders cuts in (25) and (26), dual optimal solution of the subproblems MP1-S and MP2-S should be obtained. A simulation-based approach is developed in this section. If we replace variable ε by $z - \frac{1}{TH_0(1+\Delta TH^*)}$ in MP2-S, MP2-S will be equivalent to MP1-S. Thus, we can develop one approach to derive the dual optimal solution for both MP1-S and MP2-S. Without loss of generality, we will deal with MP1-S. The dual of subproblem MP1-S denoted by MP1-S-Dual is formulated as follows:

$$max\{\sum_{i=1}^{N} \sum_{j=1}^{M} t_{i,j} u_{i,j}\}$$

$$s.t.$$

$$s_{i,j} + v_{i,j} - u_{i,j} - w_{i+b_{j-1},j-1} = 0 \quad : e^s_{i,j} \quad j = 1, ..., M,\ i = 1, ..., N,\ (i,j) \neq (1,1) \tag{27}$$

$$u_{i,j} + w_{i,j} - s_{i,j+1} - v_{i+1,j} = 0 \quad : e^d_{i,j} \quad j = 1, ..., M,\ i = 1, ..., N,\ (i,j) \neq (N,M) \tag{28}$$

$$u_{N,M} - \frac{\vartheta}{N} = 0 \quad : e^d_{N,M} \tag{29}$$

$$\alpha^s_{1,1} - u_{1,1} = 0 \quad : e^s_{1,1} \tag{30}$$

$$\vartheta = 1 \tag{31}$$

$$\alpha^s_{1,1},\ s_{i,j},\ u_{i,j},\ v_{i,j},\ w_{i,j} \geq 0 \qquad j = 1, ..., M,\ i = 1, ..., N$$

$\alpha^s_{i,j}$ and $\alpha^d_{i,j}$, except $\alpha^s_{1,1}$, are not included, because $e^s_{1,1}$ is the only event occurring at time zero, and $\alpha^s_{1,1}$ is the only one that can take non-zero value. Technically the dual variables $s_{i,j},\ v_{i,j},\ u_{i,j},\ w_{i,j}$ are defined with footnote $i = 1, ..., N$ and $j = 1, ..., M$, and in cases $i \notin \{1, ..., N\}$ or $j \notin \{1, ..., M\}$, the dual variables are equal to zero.

The explanation and proof of the proposed algorithms are based on the graph representation, so the graphs of MP1-S and MP1-S-Dual are first introduced in section 4.2.1, and the algorithms are explained accordingly in section 4.2.2.

4.2.1. Graph representation

MP1-S can also be modeled by an Event Relationship Graph (ERG), a general graphic representation of discrete event simulation (Schruben, 1983). In an ERG, each node represents an event, and each arc represents the triggering relationship between its origin and destination nodes. Figure 4 presents one element in the ERG of a serial line, which takes the starting and departure events of part i at machine j as examples. The starting event $e^s_{i,j}$ is able to be triggered after events $e^d_{i,j-1}$ and $e^d_{i-1,j}$ with zero delay, which are equivalent to constraints (21) and (22) in MP1-S, respectively. The departure event $e^d_{i,j}$ is able to be triggered after events $e^s_{i,j}$ with delay equal to $t_{i,j}$ and $e^s_{i-b_j,j+1}$ with zero delay, which are equivalent to constraints (20) and (23) in MP1-S, respectively. The entire ERG of serial production line can be found in Appendix V in online supplementary material. Besides the nodes representing events $e^s_{i,j}$ and $e^d_{i,j}$, there is also a 'start' node representing launch of the simulation run at time zero and an 'end' node representing the finish the simulation run.

MP1-S-Dual represents a weighted network flow maximization problem, whose graph is the same as the ERG of MP1-S, as mentioned in Chan and Schruben (2008). Furthermore, we introduce proposition 4.1 for detailed interpretation of MP1-S-Dual, and the proof can be found

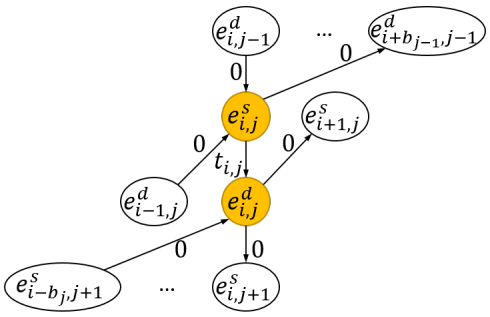


Figure 4.: Element in the ERG of serial production lines.

in Appendix IV in the online supplementary material.

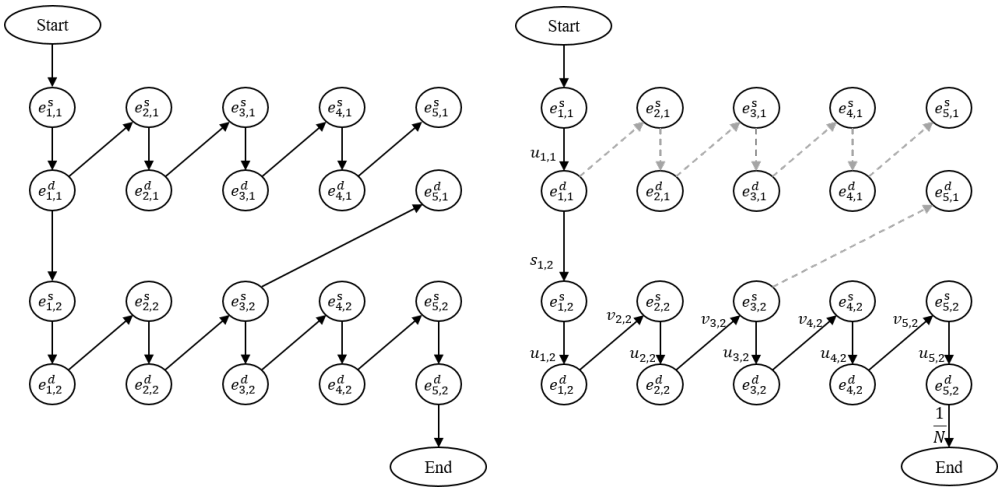
Proposition 4.1. *MP1-S-Dual is to find the maximum weighted flow in the graph of ERG. The weights on arcs are equal to the time delays, and all the nodes $e^s_{i,j}$ and $e^d_{i,j}$ hold flows. The 'start' node is a source, and the 'end' node is a sink absorbing $\frac{\vartheta}{N}$ unit flow.*

Therefore, the solution of MP1-S-Dual will be a path from the start node to the end node, the flow of all the arcs on this path will be equal to $\frac{\vartheta}{N}$, and the flow on other arcs will be equal to zero. Since the simulation model and the network flow model share the same graph, a feasible solution of MP1-S-dual can be derived from the ERG after a simulation run.

4.2.2. Cut generation

The algorithm to generate cuts is composed by two stages, i.e., simulation and dual solution calculation. A toy example in Figure 5 is first used for explanation of the procedure, the general algorithms are proposed in Algorithms 1 and 2.

First, during a simulation run, the event triggering relationships are checked, the arcs where the triggering relationship is not verified are removed, and all the remaining arcs form a subgraph of ERG, which we call *the simulated ERG*. Figure 5(a) is an example of simulated ERG of the system with $M = 2,\ b_1 = 2,\ N = 5$. Without simulation realization, the event $e^d_{5,1}$ may be triggered by $e^s_{5,1}$, or by $e^s_{3,2}$. After simulation, the constraint $e^d_{5,1} - e^s_{3,2} = 0$ is verified with equality, which means that $e^d_{5,1}$ is triggered by $e^s_{3,2}$, not $e^s_{5,1}$, and the arc between $e^s_{5,1}$ and $e^d_{5,1}$ is removed. In case one event is triggered by several events, one of them can be randomly chosen. In this way, each node has one and only one input arc, and the simulated ERG forms a spanning tree whose root is the start node. In Figure 5(b), arcs $u_{1,1},\ s_{1,2},\ u_{1,2},\ v_{2,2},\ u_{2,2},\ v_{3,2},\ u_{3,2},\ v_{4,2},\ u_{4,2},\ v_{5,2},\ u_{5,2}$ construct a path from the start node to the end node, and a feasible solution of the dual subproblem is: $u_{1,1} = s_{1,2} = u_{1,2} = v_{2,2} = u_{2,2} = v_{3,2} = u_{3,2} = v_{4,2} = u_{4,2} = v_{5,2} = u_{5,2} = \frac{\vartheta}{5} = \frac{1}{5}$, all the other variables take zero value, and constraints (27) to (31) are all satisfied.



(a) Example of simulated ERG. (b) Example of dual solution.

Figure 5.: Examples for simulation based cut generation algorithms.

Given the system parameters M and b_j, simulation parameters N and delay times $t_{i,j}$, Algorithm 1 is designed for simulating the system, and its output will be the event occurring time $e^s_{i,j}$ and $e^d_{i,j}$ and 0-1 variables $bu_{i,j}$, $bv_{i,j}$, $bw_{i,j}$, $bs_{i,j}$ representing whether the arcs are included or not in the simulated ERG. For instance, if $bu_{i,j}$ is equal to 1, arc $u_{i,j}$ remains in the simulated ERG, otherwise it is removed. Lines 5-9 of Algorithm 1 check the triggering event of each start event $e^s_{i,j}$, and lines 10-14 check the triggering event of each departure event $e^d_{i,j}$. If $e^s_{i,j}$ is triggered by $e^d_{i,j-1}$, $bs_{i,j}$ will be equal to 1, and $bv_{i,j}$ will be equal to 0; otherwise, if $e^s_{i,j}$ is triggered by $e^s_{i-1,j}$, $bv_{i,j}$ will be equal to 1, and $bs_{i,j}$ will be equal to 0. Similarly, lines 10-14 show that if $e^d_{i,j}$ is triggered by $e^s_{i,j}$, $bu_{i,j}$ will be equal to 1, and $bw_{i,j}$ will be equal to 0; otherwise, if $e^d_{i,j}$ is triggered by $e^s_{i-b_j,j+1}$, $bw_{i,j}$ will be equal to 1, and $bu_{i,j}$ will be equal to 0. The complexity of Algorithm 1 is $O(MN)$.

Given the simulated ERG, Algorithm 2 shows how to obtain the solution of MP1-S-Dual. According to constraint (29) and (31), $u_{N,M} = \frac{1}{N}$, as in lines 1. From the last event $e^d_{N,M}$ to the first event $e^s_{1,1}$, the output flow and the input flow of each node are calculated sequentially. According to (28), the input flow of node $e^d_{i,j}$ is equal to $v_{i+1,j}+s_{i,j+1}$. If $bu_{i,j} = 1$, the input flow is carried by $u_{i,j}$, so $u_{i,j} = v_{i+1,j}+s_{i,j+1}$ and $w_{i,j} = 0$; otherwise, if $bw_{i,j} = 1$, $w_{i,j} = v_{i+1,j}+s_{i,j+1}$ and $u_{i,j} = 0$. The input flow of events $e^d_{i,j}$ are calculated in lines 4–5. Similarly, the input flow of events $e^s_{i,j}$ are calculated in lines 6–7. The complexity of Algorithm 2 is $O(MN)$.

We introduce Proposition 4.2, and the proof can be found in Appendix IV in the online supplementary material.

Proposition 4.2. *The dual solution using Algorithm 1 and Algorithm 2 is optimal solution of*

Algorithm 1 Simulation of serial production lines.

Input:

$M,\ N,\ b_j,\ t_{i,j}$;

Ensure:

Event triggering relationship $bu_{i,j}, bv_{i,j}, bw_{i,j}, bs_{i,j}, \forall 1 \leq i \leq N,\ \forall 1 \leq j \leq M$.

Throughput TH

Set $bu_{i,j}, bv_{i,j}, bw_{i,j}, bs_{i,j} \leftarrow 0, \forall 1 \leq i \leq N,\ \forall 1 \leq j \leq M$
$e^s_{1,1} \leftarrow 0$
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    if $e^d_{i,j-1} > e^d_{i-1,j}$ **then**
      $e^s_{i,j} \leftarrow e^d_{i,j-1}$, $bs_{i,j} \leftarrow 1$
    else
      $e^s_{i,j} \leftarrow e^d_{i-1,j}$, $bv_{i,j} \leftarrow 1$
    end if
    if $e^s_{i,j} + t_{i,j} > e^s_{i-b_j,j+1}$ **then**
      $e^d_{i,j} \leftarrow e^s_{i,j} + t_{i,j}$, $bu_{i,j} \leftarrow 1$
    else
      $e^d_{i,j} \leftarrow e^s_{i-b_j,j+1}$, $bw_{i,j} \leftarrow 1$
    end if
  end for
end for
$TH = \frac{N}{e^d_{N,M}}$

model MP1-S-Dual.

4.3. Solution procedure

Combining BD, Algorithm 1 and Algorithm 2, the procedure to solve MP1-O and MP2-O is developed, as in Figure 6. The algorithm is initialized with the parameters and throughput of the original system in step 0, and ψ is set to $+\infty$. Step 1 calls the Algorithm 1 to simulate the system, and the exit condition is checked in step 2. The exit conditions of MP1 and MP2 are different. For MP1, the exit condition is that the solution of the master problem, i.e., ψ, in step 4 and the simulation result in step 1 converges with a tolerance ϵ_1. For MP2, the exit condition is that simulation result in step 4 satisfies the constraint (15) with a tolerance ϵ_2. Step 3 calls Algorithm 2 to get the dual optimal solution from simulation, and generates a feasibility cut (26) or an optimality cut (25). In step 4 and 5, the master problem is solved, and if it is feasible, the time to repair is updated with the master problem solution in step 6; otherwise, the original problem is infeasible and the procedure stops.

Algorithm 2 Dual optimal solution.

Input:

$M,\ N,\ b_j$;

Event triggering relationship $bu_{i,j}, bv_{i,j}, bw_{i,j}, bs_{i,j}, \forall 1 \leq i \leq N,\ \forall 1 \leq j \leq M$.

Ensure:

Dual optimal solution: $u_{i,j}, v_{i,j}, w_{i,j}, s_{i,j}, \forall 1 \leq i \leq N,\ \forall 1 \leq j \leq M$.

$\vartheta \leftarrow 1$, $\alpha^s_{1,1} \leftarrow \frac{1}{N}$, $u_{N,M} \leftarrow \frac{1}{N}$

for $i = N$ to 1 **do**

  for $j = M$ to 1 **do**

    $u_{i,j} \leftarrow bu_{i,j} * (s_{i,j+1} + v_{i+1,j})$

    $w_{i,j} \leftarrow bw_{i,j} * (s_{i,j+1} + v_{i+1,j})$

    $s_{i,j} \leftarrow bs_{i,j} * (w_{i+b_{j-1},j-1} + u_{i,j})$

    $v_{i,j} \leftarrow bv_{i,j} * (w_{i+b_{j-1},j-1} + u_{i,j})$

  end for

end for

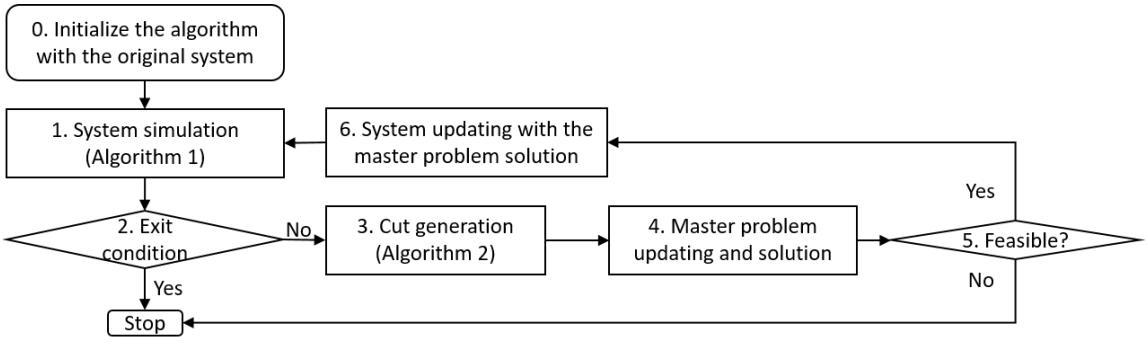


Figure 6.: The procedure to solve the throughput improvement problems.

5. Downtime bottleneck detection

5.1. Definition of downtime bottleneck on failure level

The DT-BN of the single failure mode system is defined in Chiang et al. (2000):

Definition5.1. Machine j_0 is the **DT-BN** if and only if

$$|\frac{\partial TH}{\partial T_{down,j_0}}| \geq |\frac{\partial TH}{\partial T_{down,j}}|, \forall j \neq j_0.$$

For systems with multiple failure modes, there are two ways to define the DT-BN. One is to combine multiple failure modes of each machine into a single failure mode, and using the definition above. The other way is to specify the DT-BN not only on machine level, but also on failure mode level:

Definition5.2. Failure mode k_0 of machine j_0 is the DT-BN if and only if

$$|\frac{\partial TH}{\partial T_{down,j_0,k_0}}| \geq |\frac{\partial TH}{\partial T_{down,j,k}}|, \forall j \neq j_0 \text{ or } k \neq k_0.$$

Notice that Definition 5.2 is a generalization of Definition 5.1. Using finite difference approach, the DT-BN detection problem is changed to:

$$\arg\min_{j,k}\{\Delta T_{down,j,k}|TH(\Delta T_{down,j,k}) \geq TH_0(1+\Delta TH^*)\},$$

where $TH(\Delta T_{down,j,k})$ is the expected throughput of the system where the downtime of machine j failure k is decreased by $\Delta T_{down,j,k}$, and all the other machines keep the same parameters.

5.2. DEO model for DT-BN detection

The DEO model of DT-BN detection in finite difference form is:

$$\min\{B\}$$

$$s.t.$$

$$U_{j,k}y_{j,k} \geq x_{j,k},\ \forall j=1,...,M,\ k\in\mathcal{F}_j \tag{32}$$

$$\sum_{j=1}^{M}\sum_{k\in\mathcal{F}_j} y_{j,k} = 1 \tag{33}$$

$$B = \sum_{j=1}^{M}\sum_{k\in\mathcal{F}_j}\Delta T_{down,j,k} = \sum_{j=1}^{M}\sum_{k=1}^{F_j}(T_{down,j,k} - E[f_{j,k}(x_{j,k}, r_{i,j,k,q})]) \tag{34}$$

$$y_{j,k}\in\{0,1\},\ \forall j=1,...,M,\ k\in\mathcal{F}_j$$

(3), (4), (5), (6), (7), (8), (9), (12), (13), (14), (15)

Constraints (32) show that when the downtime of failure k of machine j is reduced, the related binary variable $y_{j,k}$ should be equal to 1. Constraint (33) states that the downtime of only one failure is allowed to be reduced. Constraint (34) is a variant of (11), where $E[f_{j,k}(r_{i,j,k,q}, x_{j,k})]$ is the expected value of the reduced downtime. The cost is a linear function proportional to the expected value of the downtime reduction, while the fixed cost related to the binary variables $y_{j,k}$ is equal to zero. Specifically, if the downtime reduction function is defined as (16), then $\Delta T_{down,j,k} = x_{j,k}$. If the downtime reduction function is defined as (17), then $\Delta T_{down,j,k} = x_{j,k}(T_{down,j,k} - a_{j,k})$. If the downtime reduction function is defined as (18), then $\Delta T_{down,j,k} = x_{j,k}(T_{down,j,k} - a_{j,k} + a_{0,j,k})$. $a_{j,k}$ and $a_{0,j,k}$ play the same role as a and a_0 in (17) and (18).

It should be noticed that, for very small value of ΔTH^*, using the algorithms in Section 4 to solve the model above is equivalent to conducting sensitivity analysis over its subproblem. Chan and Schruben (2008) showed that the sensitivity estimator is an IPA estimator, as presented in

Ho, Eyler, and Chien (1983).

6. Numerical analysis

Numerical studies are conducted to analyze the algorithm performance, the throughput improvement problem and the DT-BN detection problem. Parameters of systems can be found in Appendix III in the online supplementary material. In Sections 6.1 and 6.2, the repair time follows triangular distribution. Scale downtime reduction function is used and specified as $r^{'}_{i,j,k,q} = a_{j,k} + (r_{i,j,k,q} - a_{j,k})(1 - x_{j,k})$. The cost function is linear with discontinuity at $x_{j,k} = 0$, as follows:

$$B = \sum_{j=1}^{m} \sum_{k \in \mathcal{F}_j} (C^x x_{j,k} + C^y y_{j,k})$$
$$x_{j,k} \leq U_{j,k} y_{j,k} \qquad \forall j, k$$
$$y_{j,k} \in \{0, 1\} \qquad \forall j, k,$$

where C^y is the fixed cost, C^x is the cost of unit downtime coefficient changes, and $y_{j,k}$ is the auxiliary 0-1 variables representing whether the failure is selected to improve. In Section 6.3, repair time follows exponential distribution. Scale downtime reduction function $r^{'}_{i,j,k,q} = (1 - x_{j,k}) r_{i,j,k,q}$ is used, and the cost function (34) is specified as $\sum_{j=1}^{M} \sum_{k \in \mathcal{F}_j} \Delta T_{down,j,k} = \sum_{j=1}^{M} \sum_{k \in \mathcal{F}_j} x_{j,k} T_{down,j,k}$. $U_{j,k}$, upper bound of $x_{j,k}$ is equal to 0.8. Parameters C^x, C^y, N, B^* and ΔTH^* are specified in each section. Index k is not used in single failure cases.

All the algorithms are implemented in C++. Cplex 12.5 with default settings is used to solve the master problem. Numerical study is performed on Intel(R) Xeon(R) CPU E5-2699 v4 @ 2.20GHz processor and 256GB RAM.

6.1. Algorithm performance

6.1.1. Efficiency analysis

The decrease of computational time of BD and simulation based cut generation approach compared with solving the original models MP1-O and MP2-O directly can be observed in this section. We solve MP1 and MP2 of system A-5 with $c - a = 1$, $B^* = 90$, $\Delta TH^* = 3\%$ and of system C-9 with $c - a = 2$, $B^* = 180$, $\Delta TH^* = 4\%$. Three cost function structures are analyzed, which are pure 0-1, linear and mixed integer. Pure 0-1 function is with $C^x = 0$ and $C^y = 90$,

linear function is with $C^x = 100$ and $C^y = 0$, and mixed integer function is with $C^x = 100$ and $C^y = 10$. The simulation length N is also varied, and equal to $10,000$ and $100,000$. Each system is solved with 10 sample paths and the mean computational time is reported in Table 1. The computational times for solving the problems by the proposed approach (denoted by BD+sim), by BD with subproblem solved by Cplex (denoted by BD) and by Cplex to solve the original models (denoted by Cplex) can be found in the table. In all the cases, the proposed solution approach requires the shortest computational time. To solve system C-9 with $N = 10,000$ and linear cost function, BD is slower than Cplex, implying that solving two reduced models iteratively is not as efficient as solving the original model once. However, the proposed solution approach is much faster, implying that the simulation-based cut generation approach can greatly reduce the computation burden.

			N=10,000			N=100,000		
System	Problem	Approach	0-1	Lin	Mixed	0-1	Lin	Mixed
A-5	MP1	BD+sim	0.10	5	0.28	63	96	42
		BD	52	294	51	2607	2730	2154
		Cplex	154	368	127	>10000	8671.614	>10000
	MP2	BD+sim	0.05	0.39	0.23	31	1	26
		BD	33	308	48	1559	2143	2222
		Cplex	219	431	335	>10000	9071.14	>10000
C-9	MP1	BD+sim	28	30	123	113	441	464
		BD	307	2226	2823	6697	>10000	>10000
		Cplex	3778	733	6456	>10000	>10000	>10000
	MP2	BD+sim	2	4	5	7	41	23
		BD	194	1469	632	6174	>10000	>10000
		Cplex	2475	585	2707	>10000	>10000	>10000

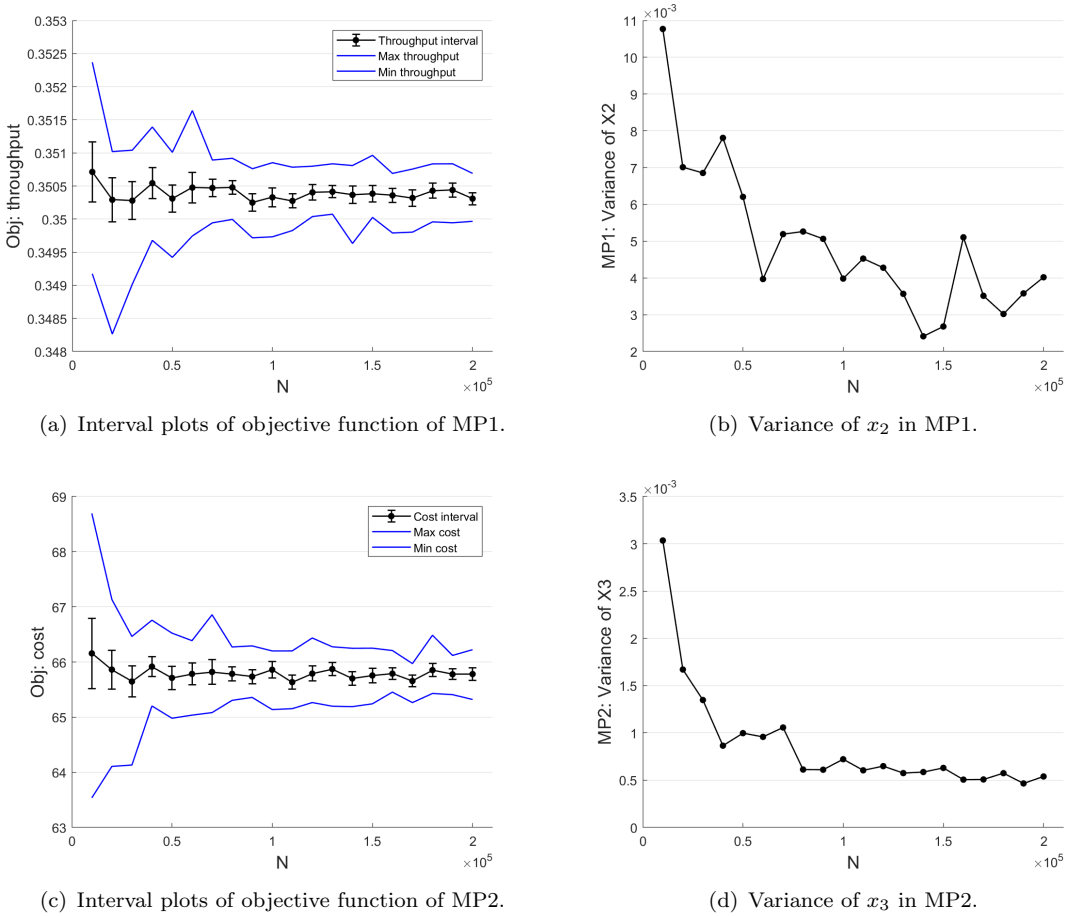
Table 1.: Mean computational time (s).

6.1.2. Convergence analysis

In this section, we show the convergence of the optimal solution as simulation length N increases. In fact, the asymptotic property of the solution of DEO as the simulation length increases has been proved theoretically within the sample path optimization framework (Robinson, 1996) in Pedrielli et al. (2018). As for the solution approach, the classic BD converges to the optimum of the original model (Benders, 1962), and the proposed simulation-based cut generation approach is exact. Thus, the convergence of the optimal solution can be guaranteed.

The experiment is conducted with system A-5, $B^* = 180$, $\Delta TH^* = 5\%$, $C^x = 100$ and $C^y = 10$. N ranges from $10,000$ to $200,000$, and each problem is replicated with 20 independent sample paths. Figure 7 shows the convergence of the solutions to MP1 and MP2, respectively.

Figure 7(a) and 7(c) include range plots and interval plots with confidence level 0.95 of the optimum, i.e., the optimal throughput and cost respectively. The range and the width of intervals become narrow as N increases. Figure 7(b) shows the variance of x_2 in MP1, which is equal to the variance of x_4, and $x_1,\ x_3,\ x_5$ are equal in all the replicates. Figure 7(d) shows the variance of x_3 in MP2, and $x_1,\ x_2,\ x_4,\ x_5$ in MP2 are equal in all the replicates. In these two plots, the variance becomes smaller as N increases. The optimal solutions based on different sample paths numerically converge as simulation length increases.



(a) Interval plots of objective function of MP1.

(b) Variance of x_2 in MP1.

(c) Interval plots of objective function of MP2.

(d) Variance of x_3 in MP2.

Figure 7.: Solution convergence of system A-5 (20 sample paths).

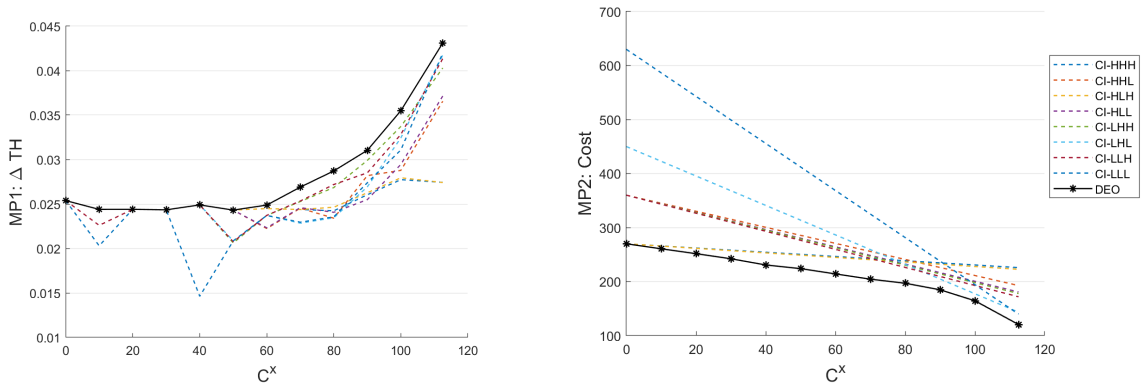
6.2. Throughput improvement problems

6.2.1. Comparison with continuous improvement

In this section, the comparison between the proposed approach and CI, the commonly used approach for manufacturing system improvement, is conducted. A system with 9 identical machines and identical buffer spaces (denoted by E-9) is used. The problems are solved with varied

unit costs C^x and C^y, while keeping the cost to improve one machine to its upper bound equal to 90, i.e., $C^y + U_{j,k}C^x = 90$. MP1 and MP2 are solved with $B^* = 270$ and $\Delta TH^* = 2\%$ respectively. CI and DEO share the same sample path of simulation length $N = 500,000$. A parametric CI algorithm is developed in Appendix II in the online supplementary material. With different parameters, 8 CIs are implemented as benchmark, denoted by CI-LLL, CI-LLH, CI-LHL, CI-LHH, CI-HLL, CI-HLH, CI-HHL, CI-HHH, and their parameters can also be found in Appendix II.

The solutions provided by the proposed method, denoted by DEO, and the 8 CIs are shown in Figure 8. The gaps between DEO and CIs exist in most of the cases. For MP1 (Figure 8(a)), the gap can be as small as zero with some values of C^x, but it becomes strictly positive for all the CIs for C^x beyond 60. For MP2 (Figure 8(b)), the gap reaches zero with only two CIs at $C^x = 0$. The results above show that the performance of CI depends on problem parameters and algorithm parameters. Thus, the selection of suitable algorithm parameters for each specific case is an important issue for CI. One advantage of the DEO approach is that there is no need to decide the algorithm parameters to obtain the optimal solution, while the solution optimality of CI cannot be guaranteed because it is a heuristic algorithm.



(a) MP1 (b) MP2

Figure 8.: Comparison with CIs.

6.2.2. Insights into throughput improvement

Five five-stage systems are analyzed in this section. System E-5 has five identical machines and buffer spaces. System B-5-I and system B-5-II are with one slow machine at the end of the line, but the buffer spaces are different. System C-5 has one slow machine in the middle and one at the end of the line. System D-5 is with two slow adjacent machines in the middle.

For system E-5, the experiment is implemented with $N = 200,000$, $C^x = 30$, $C^y = 66$ and

10 replicates. MP1 and MP2 are solved varying the values of B^* and ΔTH^*. The means of the optimal solutions are shown with color maps in Figure 9. For $B^* = 90$ and $\Delta TH^* = 1\%$, machine 3 is the only one improved. This is consistent with the bowl phenomenon of serial production lines (Rao (1976)). As B^* and ΔTH^* increases, two machines are chosen, which are machine 2 and 4, and the blockage and starvation of machine 3 are mitigated. If we further increase the budget or target, three machines, i.e., machine 2, 3 and 4, will be involved. In this case, machine 3 is improved the least, because it is relatively farther from the badly-performed machines. Similarly for the cases with $B^* = 330$ and $\Delta TH^* = 5\%$, where four machines are involved, the machine farthest from the badly-performed machine is improved the least. When all the five machines are improved, the pattern is again consistent with the bowl phenomenon, namely the downtime of the machines in the middle are reduced more.

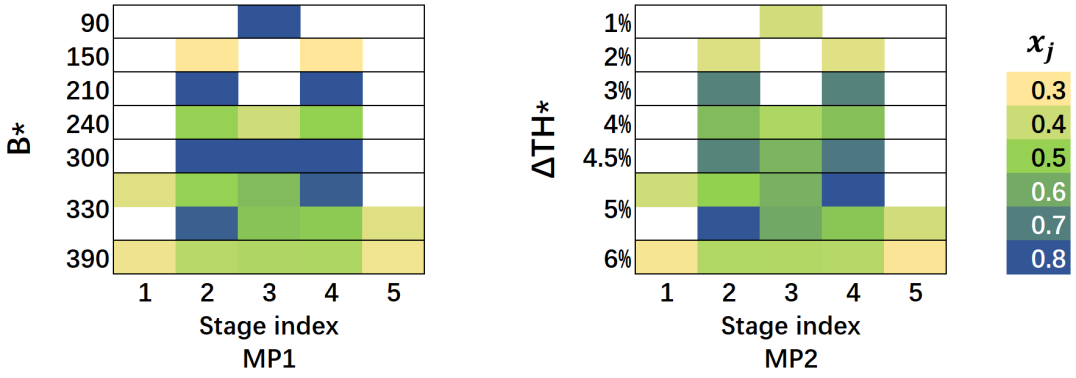


Figure 9.: Color map of optimal solutions of system E-5.

For systems B-5-I, B-5-II, C-5 and D-5, with $N = 100,000$, $C^x = 100$, $C^y = 10$, two MP1 problems and two MP2 problems are solved, denoted by MP1-L, MP1-H, MP2-L and MP2-H, respectively. The insights are addressed below, the numerical results, i.e, the optimal solution x_j, resulting throughput TH, improvement percentage $\Delta TH(\%)$ and cost can be found in Appendix VI in the online supplementary material.

In systems with one slow machine, i.e., system B-5-I and B-5-II, improving the slow machine will lead to significant throughput improvement. After the slow machine is improved to the upper bound, a secondary downtime reduction policy is reducing the downtime of its adjacent machines, but the effectiveness is not very significant. Comparing the results of system B-5-I and B-5-II, the buffer sizes affect optimal solution. System B-5-II has more buffer spaces in front of the slow machine, so it is more decoupled from the upstream machines. Consequently, with budget equal to 180, system B-5-II allocates budget only to the last two machines but system B-5-I allocates budget to the last three machines. Furthermore, for system B-5-II, increasing

budget from 90 to 180, the throughput improvement percentage is almost stable from 8.12% to 8.13%, but the percentage is increased from 7.76% to 8.34% for system B-5-I. Thus, changing buffer allocation leads to different downtime reduction policies and their performance.

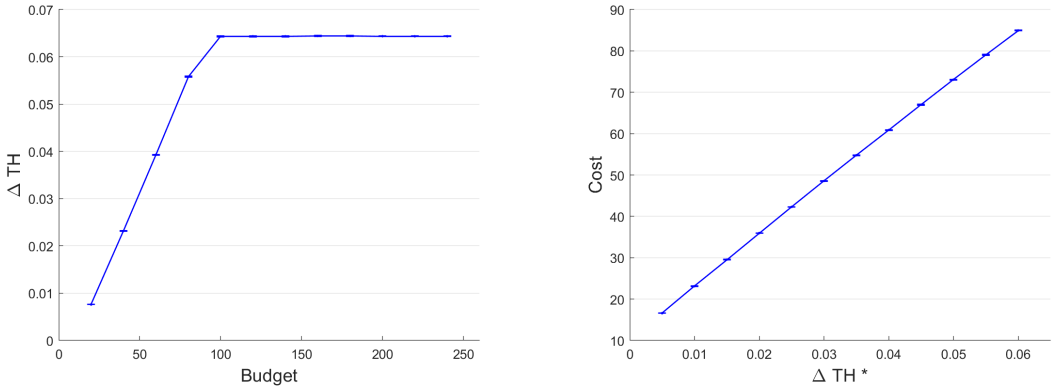
For systems with two slow machines, i.e., systems C-5 and D-5, the two slow machines are improved simultaneously. However, the amount of downtime reduction, i.e., x_j, are different for the two systems. For system C-5, downtimes of the two slow machines are reduced with similar amount, but in system D-5, the machine in the middle is improved more. Thus, for systems with the same machine parameters and buffer allocation, the position of the slow machines affects the optimal downtime reduction amount of each machine.

6.2.3. Long line case

We apply the proposed approach to a 23-stage line, which is a real life case introduced in Tempelmeier (2003). For the long line, we solve the MP1 with different values of budget B^*, ranging from 20 to 240, and each is solved with ten sample paths. The interval plot in Figure 10(a) shows how the throughput increment percentage varies as budget increases. For budget from 20 to 80, ΔTH increases linearly, and reaches its limit 6.4% for $B^* \geq 100$. Similarly, MP2 is solved with target ΔTH^* ranging from 0.5% to 6%, and each is solved with ten sample paths. The interval plot in Figure 10(b) shows that the cost increases linearly for $\Delta TH^* \leq 6\%$, and the cost to achieve $\Delta TH^* = 6\%$ is equal to 84.9. The problem becomes infeasible for $\Delta TH^* \geq 6.5\%$. For both MP1 and MP2, throughput improvement is effective where the curves show linear pattern, and the relevant solution is to improve the machine 21 only. After improving machine 21 to the best, i.e., x_{21} reaching its upper bound, it will not be beneficial to improve any other machines, since the system throughput is still highly limited by the performance of machine 21. The experiment of MP2 with $B = 240$ requires the longest computational time, which is less than 765 seconds with simulation length $N = 5,000,000$.

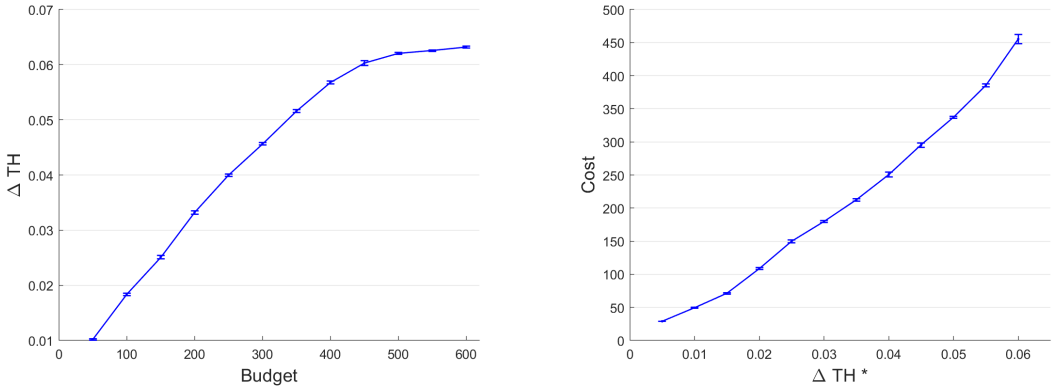
6.2.4. Multi-failure case

We consider a line where all machines are with 2 failure modes. Type 1 failure refers to less frequent failure requiring long time to repair, and type 2 refers to frequent failure requiring short time to repair. We solve MP1 with budget B^* ranging from 50 to 600, and each is solved with ten sample paths. With $\Delta TH^* \leq 2\%$ and $B^* \leq 100$, the type 1 failure of machine 3 is improved first, and the improvement of type 1 failure of machines 2 and 8 and type 2 failure of machine 2 become relevant when B^* is increased to 400 and ΔTH^* is increased to 5%. The interval plot



(a) MP1. (b) MP2.

Figure 10.: Interval plots of solutions of the long line (10 sample paths).



(a) MP1. (b) MP2

Figure 11.: Interval plots of solutions with different budget or target of the multi-failure system (10 sample paths).

in Figure 11(a) shows that the throughput increases significantly through the entire curve, but the slope becomes flatter as B^* increases. Similarly, MP2 are solved with different values of target ΔTH^*, ranging from 0.5% to 6%, and each is solved with ten sample paths. The interval plot in Figure 11(b) shows that cost increases faster as ΔTH^* increases, since more failures are involved.

6.3. DT-BN detection: comparison with the arrow method

In this section, the numerical results of the downtime bottleneck detection is presented. The approach proposed in this work (denoted by DEO) is compared with the Arrow Method (AM). Three systems, DT-BN 1, DT-BN 2 and DT-BN 3, are investigated. DT-BN 1 is used in Chiang et al. (2000), where AM fails to detect the bottleneck, DT-BN 2 is used in Yu and Matta (2016)

and Betterton and Silver (2012) as a troublesome case, and DT-BN 3 is a multi-failure line adapted from DT-BN 2. For the DEO approach, the parameter ΔTH^* is equal to 0.001. The bottleneck of DT-BN 1 and DT-BN 2 is machine 5 and machine 3, respectively. The bottleneck of DT-BN 3 at machine level and failure level is machine 3 and failure type 1 of machine 6, respectively. The performance indicator is the correct detection probability, calculated as the frequency of correct detection over 100 replicates.

The results of systems DT-BN 1 and DT-BN 2 are shown in Figure 12. Since both AM and DEO take information from simulation, the correct detection probability is affected by simulation length, and the correct detection probability will converge to one or zero as simulation length increases. For system DT-BN 1, the correct detection probability of AM is down to 0 at $N = 50,000$, and the correct detection probability of DEO is up to 1 at $N = 8,000,000$. For system DT-BN 2, AM correctly detects the bottleneck in all the replicates with $N = 100,000$, whereas DEO requires a much longer simulation with $N = 5,000,000$. System DT-BN 3 is a multi-failure system in which the machine level bottleneck is different from the failure level. The failure level bottleneck can be correctly detected by DEO in all the replicates with $N = 100,000$. However, AM can handle bottleneck detection only at machine level, and provides machine 3 as result.

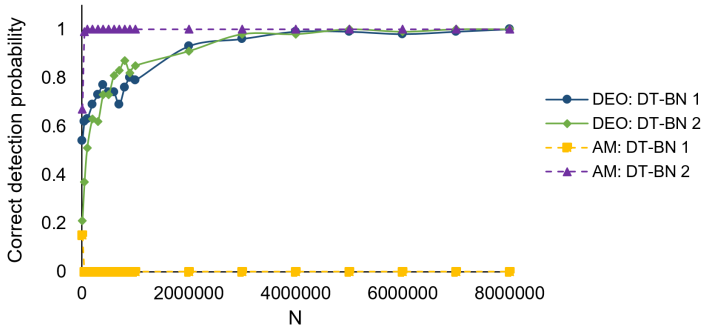


Figure 12.: Correct detection probability of DEO and AM.

In conclusion, the correct detection probability of DEO numerically converges to one as sample path length increases, but the convergence rate might be slow for difficult cases. The solution of AM may converge faster but it may provide wrong results, since it is heuristic. DEO is able to handle downtime bottleneck detection at failure level for multi-failure serial line.

7. Conclusion

In this work, we propose MILP models for the throughput improvement problem of serial production lines via downtime reduction. Two problems are defined and solved. One is to achieve maximal throughput improvement within a given budget, and the other is to achieve a fixed throughput improvement with the minimal cost. The solution of the problems includes a set of failures to be improved and the amount of the downtime reduction of each failure. MILP models are developed using the samples of processing time, uptime and downtime, and the data can be either generated random variate or collected field data. An algorithm based on Benders decomposition and discrete event relationships of serial production lines is proposed to solve the MILP models. Numerical analysis shows that the proposed solution algorithm can significantly save the computational time, and that solutions of a problem based on different sample paths converge as sample size increases. In addition, the definition of downtime bottleneck is generalized to failure level to deal with multi-failure systems. The proposed models and algorithms are applied to several cases, including a 23-stage practical line and a multi-failure line, and some management insights are also derived. Comparison with CI and AM, two state-of-the-art heuristic approaches, shows that the proposed approach leads to solution with better objective functions and higher level of bottleneck detection correctness. This shows a major contribution of this work, which is the formal representation and sample path based exact solution approach of the throughput improvement problem and the downtime bottleneck detection. Thus, it can be used as a benchmark for other algorithms.

The simulation-based cut generation approach proposed in this work is a primary step for using information in the simulation trajectory to solve optimization problems of discrete event systems. How to generate cuts from simulation trajectory of generic discrete event systems, and how to effectively embed the obtained information in various optimization methods are of future research interest.

Funding

This work is supported by the National Natural Science Foundation of China under grants 61473188.

References

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. Numerische mathematik, 4(1), 238–252.

Betterton, C., & Silver, S. (2012). Detecting bottlenecks in serial production lines–a focus on interdeparture time variance. International Journal of Production Research, 50(15), 4158–4174.

Chan, W. K., & Schruben, L. (2008). Optimization models of discrete-event system dynamics. Operations Research, 56(5), 1218–1237.

Chiang, S.-Y., Kuo, C.-T., & Meerkov, S. M. (2000). Dt-bottlenecks in serial production lines: theory and application. IEEE Transactions on Robotics and Automation, 16(5), 567–580.

Colledani, M., Ekvall, M., Lundholm, T., Moriggi, P., Polato, A., & Tolio, T. (2010). Analytical methods to support continuous improvements at scania. International Journal of Production Research, 48(7), 1913–1945.

Fu, M. C., et al. (2015). Handbook of simulation optimization (Vol. 216). Springer.

Ho, Y., & Cao, X. (1983). Perturbation analysis and optimization of queueing networks. Journal of Optimization Theory and Applications, 40(4), 559–582.

Ho, Y., & Cao, X. (2012). Perturbation analysis of discrete event dynamic systems (Vol. 145). Springer Science & Business Media.

Ho, Y., Eyler, M., & Chien, T. (1983). A new approach to determine parameter sensitivities of transfer lines. Management Science, 29(6), 700–714.

Kang, N., Zhao, C., Li, J., & Horst, J. A. (2016). A hierarchical structure of key performance indicators for operation management and continuous improvement in production systems. International journal of production research, 54(21), 6333–6350.

Li. (2009). Bottleneck detection of complex manufacturing systems using a data-driven method. International Journal of Production Research, 47(24), 6929–6940.

Li, J. (2013). Continuous improvement at toyota manufacturing plant: applications of production systems engineering methods. International Journal of Production Research, 51(23-24), 7235–7249.

Li, J., & Meerkov, S. M. (2008). Production systems engineering. Springer Science & Business Media.

Matta, A. (2008). Simulation optimization with mathematical programming representation of discrete event systems. In Proceedings of the 2008 winter simulation conference (pp. 1393–1400). Miami, Florida.

Pedrielli, G., Matta, A., & Alfieri, A. (2015). Integrated simulation-optimization of pull control systems. International Journal of Production Research, 53(14), 4317–4336.

Pedrielli, G., Matta, A., Alfieri, A., & Zhang, M. (2018). Design and control of manufacturing systems: a discrete event optimisation methodology. International Journal of Production Research, 56(1-2), 543–564.

Rao, N. P. (1976). A generalization of the 'bowl phenomenon'in series production systems. The International Journal of Production Research, 14(4), 437–443.

Robinson, S. M. (1996). Analysis of sample-path optimization. Mathematics of Operations Research, 21(3), 513–528.

Schruben, L. (1983). Simulation modeling with event graphs. Communications of the ACM, 26(11), 957–963.

Tan, B. (2015). Mathematical programming representations of the dynamics of continuous-flow production systems. IIE Transactions, 47(2), 173–189.

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

Weiss, S., & Stolletz, R. (2015). Buffer allocation in stochastic flow lines via sample-based optimization with initial bounds. OR Spectrum, 37(4), 869–902.

Yu, C., & Matta, A. (2016). A statistical framework of data-driven bottleneck identification in manufacturing systems. International Journal of Production Research, 54(21), 6317–6332.

Zhang, M., Matta, A., Alfieri, A., & Pedrielli, G. (2017, August). A simulation-based benders' cuts generation for the joint workstation, workload and buffer allocation problem. In Conference on automation science and engineering, 2017. Xi'an, China.

Zhang, M., Matta, A., & Pedrielli, G. (2016). Discrete event optimization: Workstation and buffer allocation problem in manufacturing flow lines. In Proceedings of the 2016 winter simulation conference (wsc) (pp. 2879–2890). Washington, D.C.: Institute of Electrical and Electronics Engineers, Inc.

Notes on contributors

Mengyi Zhang is PhD candidate of Department of Mechanical Engineering at Politecnico di Milano, Italy. She received her B.E. degree in mechanical engineering in 2014 and M.Sc. degree in industrial engineering in 2017, in Shanghai Jiao Tong University, China. Her research interests include simulation optimization of manufacturing systems.

Andrea Matta is Professor of Department of Mechanical Engineering at Politecnico di Milano, where he currently teaches integrated manufacturing systems and manufacturing. His research area includes analysis and design of manufacturing and health care systems. He is Editor-in-Chief of Flexible Services and Manufacturing Journal.