Optimization
Vol. 00, No. 00, Month 200x, 1–24

Production Systems with Limited Repair Capacity

S. Göttlich[a], M. Herty[b], C. Ringhofer[c] and U. Ziegler[b]

[a] *TU Kaiserslautern, Department of Mathematics, Postfach 3049, 67653 Kaiserslautern, Germany*; [b] *Lehrstuhl C für Mathematik, Templergraben 55, 52062 Aachen, Germany*;[c] *Department of Mathematics and Statistics Arizona State University, Tempe, AZ 85287-1804, USA*

(*Received 00 Month 200x; in final form 00 Month 200x*)

Optimizing manufacturing systems consists in generating large-quantity outputs to fulfill customers demands. But naturally machines may fail and the production process is either slowed down or completely interrupted. In order to keep production running, we are interested in assigning repair crews to currently broken-down machines. But due to the limited repair capacity and the dynamics involved in the production process, we propose a scheduling problem based on ordinary differential equations for the description of buffer levels and the actually available processing capacity. We discuss properties of the model and present a solution approach leading to a mixed-integer programming model.

Keywords: scheduling, production networks, combinatorial optimization, network flows

AMS Subject Classification: 90B10, 49M25

1. Introduction

A production system usually comprises a large number of suppliers, storage units and machines arranged in a network structure. During the last years several dynamic production models have been developed either describing the trajectory of each good through the network, which is called discrete event simulation [2], or by using so-called fluid models models, where averaged quantities are used to track goods, see [1–4, 7, 9, 13] for an overview. Based on these continuous production models, optimization problems have been introduced [8, 10, 11, 14]. These models typically deal with economic issues such as minimization of costs or maximization of output and allow for the interpretation of optimal routing of goods through a manufacturing system, best possible production mix or optimal order policies.

A further application of dynamic optimization problems are allocation problems where the underlying dynamics are given by a set of ordinary differential equations (ODEs). Think of an interconnected network of machines whose current load is simply given by an equation measuring the ratio between influx and outflux. In best case, all machines are running and work with maximal capacity. Otherwise it may happen that machines fail due to wear or a break. Then a capacity loss will occur. The aim is now to assign repair workers to the broken machines to maintain production and in particular, to increase the processing capacities again. However, this will lead to a great dilemma since repair capacity is limited.

[a]Corresponding author. Email: goettlich@mathematik.uni-kl.de

ISSN: 0233-1934 print/ISSN 1029-4945 online

DOI: 10.1080/0233193YYxxxxxxxx
http://www.informaworld.com

To solve this ODE-restricted optimal control problem, we set up an optimization problem that provides an optimal schedule for repair workers that might vary in time, see [19]. Major application areas for scheduling problems are airline crew scheduling, course timetable scheduling, flow shop and job shop scheduling, as for instance [6, 15] and references therein. The purpose of scheduling problems is to assign existing resources and to ensure fairness amongst the users utilizing the resources. Normally, the scheduling is a static process providing a fixed shift assignment. This is different to our model since repair workers are allowed to change location within a predefined time period. Therefore, we propose a mixed integer problem, as originally introduced in [8], that enables us to take advantage of MIP solvers using branch and cut-algorithms. In doing so we emphasize in particular the benefits of a dynamic modeling approach and show how the repair worker switching influences the maximal throughput in the system.

The outline of the paper is as follows: In Section 2, we introduce a mass-conserving network model including equations for the processor load and the capacity drop as well. We explain the role of temporary worker assignments and apply a suitable numerical discretization to solve the underlying optimization problem. Section 3 is devoted to a steady state analysis and the connection to the well known maximum flow problems from graph theory. Computational experiments are provided in Section 4. They are used to discuss the aforementioned phenomena.

2. Repair scheduling

Our modeling approach is motivated by failures or limited productivity of operating processors. To keep the model simple, reliability/breakdown rates are integrated into the model using experience values to avoid the impact of stochasticity. Anyway, the real problem lies in maintaining the current production process. Thus, we focus on a strategy aimed at maximum output while cleverly allocate a limited number of repair workers.

2.1. *Model description*

We start this paragraph with the common definition of a production network. The latter is described by a directed graph $G = (V, E)$, where V denotes the set of vertices and E the set of edges. The subset E_{out} contains the edges, after which goods leave the system. They are called outflow edges. Furthermore, E_{in} comprehends all edges, where external inflow (e.g. raw material) is introduced into the system. In the same way all starting vertices of inflow edges are denoted by V_{in} and terminal nodes of outflow edges by V_{out}.

Each edge is associated with an assembly line where the flow of goods is transported from one machine to another and can be piled up in front of machines, cf. Figure 1. The assembly line and the storage are treated as one entity and will be simply referred to as *buffer* in the course of this article. The variable $u_i(t)$ denotes the number of parts in the buffer i at time t and τ_i denotes the constant throughput time. The flow of goods passing a machine is given by $f_i(t)$ that is bounded from above by the processing capacity $c_i(t)$ (measured in parts per unit time). Different from former production network models such as [9, 14], the capacity is not a fixed parameter, but can fluctuate within the production process. There is a fixed upper bound for $c_i(t)$, the maximal capacity $\mu_i > 0$. The evolution of capacities depends on the constant breakdown rates α_i, the constant repair rates d_i and the percentage of repair workers allocated to each machine, denoted by $\beta_i(t) \in [0, 1]$. The total

number of repair workers is given by W. Network information and the distribution of flow are coded in the matrix B.

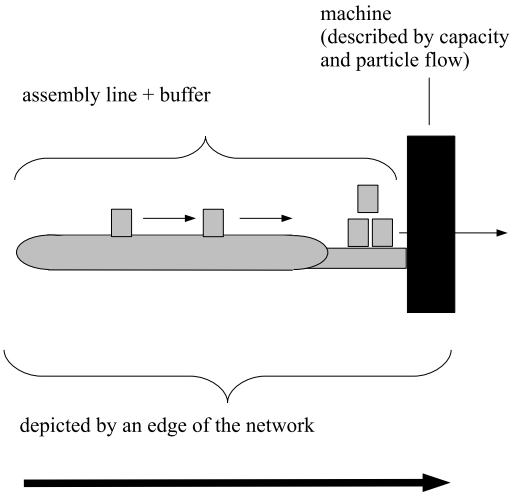


Figure 1. Sketch of processor layout.

The constant matrix entries B_{ij} denote the percentage of flow going from edge j to edge i. We have the total ingoing flow into buffer i at time t given by $\sum_{j\in E} B_{i,j} f_j(t)$ (shortly $Bf(t)$) where $B_{ii} = 0$. Kirchhoff's law and conservation of mass through nodes imply that the column sums are $\sum_{i\in E} B_{i,j} = 1$ except for all outgoing edges $j \in E_{out}$. In case of more than one outgoing processor, there is freedom in partitioning the flow of goods, cf. Figure 2. Otherwise, in case of just one outgoing processor, the entries of the matrix B will be 1. Hence, the transpose B^T with its entries $B_{ij} > 0$ is the connectivity matrix of the network. An example is shown in Figure 2.

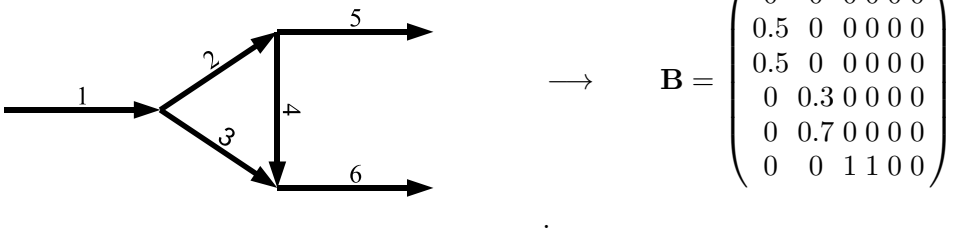


Figure 2. Example network with corresponding matrix $\mathbf{B}$.

Now, we are able to consider equations describing the dynamics of the buffer level u_i at time t. This linear rate equation measures the difference between the in- and outgoing flow of goods. External inflow into processor i will be prescribed by $f_{ext,i}(t)$ for $i \in E_{in}$. For all other processors $f_{ext,i}(t) = 0$ will hold. Since in a directed graph, the flow direction is fixed, i.e. the flow entering a processor is $Bf(t)$, we have to think about a rule to determine the outgoing flux $f_i(t)$. As proposed in [9, 10], we assume

$$f_i(t) = \begin{cases} \min\{Bf(t), c_i(t)\}, & u_i(t) = 0 \\ c_i(t), & u_i(t) > 0 \end{cases} \tag{1}$$

distinguishing the two states empty or non-empty buffers. Introducing a small relaxation parameter τ_i (in our case this is the throughput time), equation (1) can be rewritten as

$$f_i(t) = \min\Big\{c_i(t), \frac{u_i(t)}{\tau_i}\Big\}, \tag{2}$$

where $\frac{u_i(t)}{\tau_i} \approx Bf(t)$, see details in [1, 14]. In summary, we consider the following ordinary differential equation for the buffer levels:

$$\partial_t u_i(t) = Bf(t) + f_{ext,i}(t) - f_i(t) \tag{3a}$$

$$f_i(t) = \min\Big\{c_i(t), \frac{u_i(t)}{\tau_i}\Big\}. \tag{3b}$$

The interpretation of (3) is as follows: Goods are fed into the network, flow from one machine to another and can be stored in buffers in case of capacity shortage. As soon as goods have traversed the outflow edges, they will leave the system. It is intuitively clear, that the conservation of mass through the whole network should hold, since no goods are lost or generated inside the network. This gives rise for the following lemma.

Lemma 2.1: *Let $t \in [0,T] \subset \mathbb{R}$ and $f_i : [0,T] \to \mathbb{R}_0^+$ and $f_{ext,i} : [0,T] \to \mathbb{R}_0^+$ be L^1 functions for all i. Then the total conservation of mass for an initially empty network, i.e. $u_i(0) = 0$, is given by*

$$\int_0^t \sum_{i\in E} f_{ext,i}(\tilde{t})\, d\tilde{t} = \sum_{i\in E} u_i(t) + \int_0^t \sum_{i\in E_{out}} f_i(\tilde{t})\, d\tilde{t} \quad \forall t \in [0,T], \tag{4}$$

That means the total number of incoming goods until a certain time t has to be equal to the number of goods, that remain inside the network, i.e. stored in buffers, at time t plus the goods, that have already left the network.

Proof: We argue, that the total conservation of mass holds true due to constraint (3) which ensures the conservation of mass through nodes only. From the construction of the matrix B, we know that each column, that represents an edge which is not an outflow edge, contains positive entries between zero and one, describing how much percent of the flow is sent to consecutive processors. Since flow is distributed between consecutive edges, we know $\sum_{i\in E} B_{i,j} = 1$ except for all outgoing edges $j \in E_{out}$. Starting from the total flow in the network and considering the structure of B, we deduce for all t:

$$\sum_{i\in E}(B\cdot f(t))_i = \sum_{i\in E}\sum_{j\in E} B_{ij} f_j(t) = \sum_{j\in E}\Big(\underbrace{\sum_{i\in E} B_{ij}}_{=\begin{cases}1\ \forall j \notin E_{out}\\ 0 \text{ else}\end{cases}}\Big) f_j(t) = \sum_{i\in E\setminus E_{out}} f_i(t). \tag{5}$$

Next, we take equation (3), integrate both sides with respect to time and take the

sum over all processors:

$$\sum_{i\in E} u_i(t) - \sum_{i\in E} u_i(0) = \int_0^t \Big(\underbrace{\sum_{i\in E} (B\cdot f(\tilde{t}))_i}_{\text{see (5)}} - \sum_{i\in E} f_i(\tilde{t}) + \sum_{i\in E} f_{ext,i}(\tilde{t}) \Big)\, d\tilde{t}$$

$$\sum_{i\in E} u_i(t) - \sum_{i\in E} u_i(0) = \int_0^t \Big(\underbrace{\sum_{i\in E\setminus E_{out}} f_i(\tilde{t}) - \sum_{i\in E} f_i(\tilde{t})}_{=-\sum_{i\in E_{out}} f_i(\tilde{t})} \Big)\, d\tilde{t} + \int_0^t \sum_{i\in E} f_{ext,i}(\tilde{t})\, d\tilde{t}$$

Sorting all terms and inserting the initial conditions $u_i(0) = 0\,\forall i$ yields the desired result. □

The next modeling step is more involved. We need a dynamic equation for the capacity $c_i(t)$ which covers the failure of machines with the available repair capacity. The main ingredients are the following scenarios:

- The machine is running and works with maximal capacity μ_i. We have no capacity loss and no workers have to be assigned.
- Breakdown of machines: The maximum capacity will be reduced and workers need to be assigned.

According to the ideas before, we derive a rate equation depending on a deterministic breakdown rate $\alpha_i > 0$ and the term $Wd_i\beta_i(t)$ characterising the percentage of assigned workers. Here, the factor d_i is solely used to get the right scaling (since $c_i(t)$ is measured in parts per unit time).

$$\partial_t c_i(t) = \min\Big\{\frac{\mu_i - c_i(t)}{\epsilon}, Wd_i\beta_i(t)\Big\} - \min\Big\{\frac{c_i(t)}{\epsilon}, \alpha_i\Big\}, \quad \epsilon \ll 1. \tag{6}$$

The interpretation is as follows (performing a one-to-one comparison with (1) and (2)):

1. Assume we have a total loss of capacity, i.e. $c_i(t) = 0$. This implies either $\partial_t c_i(t) = \frac{\mu_i}{\epsilon}$ or $\partial_t c_i(t) = Wd_i\beta_i(t)$. In both cases the broken machine will be repaired and the capacity starts increasing again.
2. The machine works with maximal capacity $\mu_i = c_i(t) > 0$. Equation (6) reduces to $\partial_t c_i(t) = -\alpha_i$. Then, the capacity rate can be only decreased.
3. The capacity is $0 < c_i(t) < \mu_i$. This yields $\partial_t c_i(t) = Wd_i\beta_i(t) - \alpha_i$. In-and decreasing capacity might be possible.

To get a well-defined allocation of the available workers, we state the following condition on the decision variable $\beta_i(t)$:

$$\sum_{i\in E} \beta_i(t) = 1, \quad 0 \le \beta_i(t) \le 1, \qquad \forall i, t. \tag{7}$$

There are some remarks in order:

Remark 1 :

- Since time-dependent parameters $\beta_i(t)$ which allow for a worker change in each time are not convenient in practice, we choose β_i either to be constant or piecewise constant, see Section 4.

- Workers are assigned simultaneously and immediately, i.e. time delays between different machines are neglected.

The complete optimization problem we are interested in consists of maximizing the total outflow of the network (8a) subject to the buffer level equation (3), the capacity drop (6) and the restriction for the assignment parameters (7). Obviously, we end up with an ODE-restricted optimization problem where control variables are $\beta_i(t)$, i.e. the assignment of the repair workers to the machines. Hence, the entire optimization problem reads $\forall i, t$:

$$\max \int_0^T \sum_{i \in E_{out}} f_i(t) dt \tag{8a}$$

s. t.

$$(3), (6), (7) \tag{8b}$$

$$u_i(0) = u_{0i}, \quad c_i(0) = c_{0i} \tag{8c}$$

$$0 \leq \beta_i(t) \leq 1 \tag{8d}$$

$$0 \leq c_i(t) \leq \mu_i, \quad u_i(t) \geq 0. \tag{8e}$$

Constraint (8c) defines the initial conditions for the corresponding ODEs. If not said otherwise, we choose as initial condition empty buffers (i.e. $u_i(t) = 0$) and full capacities (i.e. $c_i(t) = \mu_i$). Finally, constraint (8e) represents additional non-negativity and box conditions.

2.2. *Mixed integer approach*

One way to solve the ODE-restricted optimization problem (8) is to use a linear mixed-integer programming model (MIP). The latter can be obtained employing common numerical discretizations combined with rewriting techniques borrowed from discrete optimization. It is in fact possible to convert particular nonlinear structures (e.g. the *min*-function) into a dynamic mixed-integer framework. This alternative has been originally introduced in [8] and has been successfully applied to a wide variety of production problems in the meanwhile, see [11, 14].

First of all, we choose a uniform discrete time grid $\mathcal{T} = \{t : t = 0, \ldots, n_t\}$ of the underlying interval $[0, T]$ where n_t denotes the number of grid points. The step-size is defined via $\Delta t = \frac{T}{n_t}$.

Then, in a first step, in a straightforward manner, both ordinary differential equations (3) and (6) are discretized using the explicit Euler scheme.

Remark 2 : For the step-size Δt, the condition

$$\Delta t := \min\{2\tau_i, 2\epsilon\}, \quad \forall i \in E$$

must be satisfied since we deal with stiff problems. An implicit discretization is avoided, since this will lead to an intractable problem during the solution procedure.

As a preliminary result, we get the following optimization problem $\forall i, t \in \mathcal{T}$:

$$\max \sum_{i \in E_{out}} \sum_{t \in \mathcal{T}} f_i^t \cdot \Delta t$$

s. t.

$$u_i^{t+1} = u_i^t + \Delta t \cdot [[B \cdot f]_i + f_{ext,i}^t - f_i^t] \tag{9a}$$

$$c_i^{t+1} = c_i^t + \Delta t \cdot [D_i^t - A_i^t] \tag{9b}$$

$$\sum_{i \in E} \beta_i^t = 1 \tag{9c}$$

$$u_i^0 = u_{0i}, \quad c_i^0 = c_{0i} \tag{9d}$$

$$0 \leq \beta_i^t \leq 1 \tag{9e}$$

$$0 \leq c_i^t \leq \mu_i, \quad u_i^t \geq 0, \qquad , \tag{9f}$$

with $A_i^t := \min\{\frac{c_i^t}{\epsilon}, \alpha_i\}$, $D_i^t := \min\{\frac{\mu_i - c_i^t}{\epsilon}, W d_i \beta_i^t\}$ and $f_i^t := \min\{c_i^t, \frac{u_i^t}{\tau_i}\}$. This looks quite similar to (8) where (9c) describes the worker distribution rates, (9d) the initial conditions and (9e)-(9f) represent box constraints.

Now, the crucial point in order to get a linear MIP is to linearize the min-terms in A_i^t, D_i^t and f_i^t with respect to c_i^t, u_i^t and β_i^t. We make use of the following proposition:

Proposition 2.2: *In general terms, an expression of the form $c = \min\{a, b\}$ can be linearized by introducing a binary variable $\gamma \in \{0, 1\}$ and using the additional unequality constraints*

$$\gamma \cdot a \leq c \leq a$$
$$b - K \cdot \gamma \leq c \leq b$$

where K is sufficiently large, such that $K > b$ holds.

One can easily check, that $\gamma = 1$ is equivalent to the case $c = a$, and $\gamma = 0$ is valid, if and only if $c = b$, cf. references [8, 17]. In this way, due to Proposition 2.2, we get for A_i^t the following constraints

$$\alpha_i \cdot \kappa_i^t \leq A_i^t \leq \alpha_i \tag{10a}$$

$$\frac{c_i^t}{\epsilon} - \overline{M} \cdot \kappa_i^t \leq A_i^t \leq \frac{c_i^t}{\epsilon}, \tag{10b}$$

where $\overline{M} := \frac{\mu_i}{\epsilon}$ and κ_i^t is binary, i.e. $\kappa_i^t \in \{0, 1\}$. By applying the same method to f_i^t, we end up with

$$g_i^t \leq f_i^t \leq c_i^t \tag{11a}$$

$$\frac{u_i^t}{\tau_i} - M \xi_i^t \leq f_i^t \leq \frac{u_i^t}{\tau_i}, \tag{11b}$$

where M is a sufficiently large constant, ξ_i^t are additional binary variables and

$$g_i^t := c_i^t \cdot \xi_i^t. \tag{12}$$

However, we are not done at this point, since $c_i^t \cdot \xi_i^t$ is a product of two unknowns, and thus linearity is not provided. To get rid of this drawback, we use another technique. We treat g_i^t as real variables and describe (12) by

$$0 \leq g_i^t \leq \mu_i \xi_i^t \tag{13a}$$

$$c_i^t - \mu_i(1 - \xi_i^t) \leq g_i^t \leq c_i^t. \tag{13b}$$

Following Proposition 2.2 and taking computational runtime into account, we need an estimate for the constant M in (11). More precisely, it is important to choose M as tight as possible. Hence, let M depend on i and t and make sure, that

$$M_i^t \geq \frac{u_i^t}{\tau_i}, \tag{14}$$

holds $\forall i$, $t \in \mathcal{T}$. Therefore, we consider (9a) in order to derive an upper bound for u_i^t:

$$u_i^t \leq u_i^{t-1} + \Delta t \cdot [B \cdot \mu]_i + \Delta t \cdot f_{ext,i}^{t-1},$$

where μ denotes a vector-valued function with entries μ_i for each machine. From our initial conditions we know, that $u_i^0 = 0$. Hence, we iteratively get

$$u_i^t \leq t \cdot \Delta t \cdot [B \cdot \mu]_i + \Delta t \cdot \sum_{\bar{t}=0}^{t-1} f_{ext,i}^{\bar{t}}$$

and

$$M_i^t := \frac{1}{\tau_i} t \cdot \Delta t \cdot [B \cdot \mu]_i + \frac{1}{\tau_i} \Delta t \cdot \sum_{\bar{t}=0}^{t-1} f_{ext,i}^{\bar{t}} \tag{15}$$

respectively.

Remark 3: As beneficial side-effect, we have gained an upper bound for u_i^t, namely

$$0 \leq u_i^t \leq \tau_i \cdot M_i^t, \quad \forall i,\, t \in \mathcal{T}. \tag{16}$$

Note that it is advantageous to keep box constraints as tight as possible, since this might lead to smaller branch and bound trees and resulting runtime reductions.

The missing linearization of D_i^t is now done analogeously. This leads to the following additional constraints:

$$W d_i \cdot h_i^t \leq D_i^t \leq W d_i \beta_i^t \tag{17a}$$

$$\frac{\mu_i - c_i}{\epsilon} - \frac{\mu_i}{\epsilon} \cdot \gamma_i^t \leq D_i^t \leq \frac{\mu_i - c_i^t}{\epsilon} \tag{17b}$$

$$0 \leq h_i^t \leq \gamma_i^t \tag{17c}$$

$$\beta_i^t - (1 - \gamma_i^t) \leq h_i^t \leq \beta_i^t \tag{17d}$$

where $\gamma_i^t \in \{0, 1\}$ and h_i^t is a new set of real variables describing the nonlinearity $\beta_i^t \cdot \gamma_i^t$.

Finally, we specify box and binary constraints for all new variables:

$$0 \leq h_i^t \leq 1,\ 0 \leq g_i^t \leq \mu_i, \tag{18a}$$

$$0 \leq f_i^t \leq \mu_i,\ 0 \leq A_i^t \leq \alpha_i,\ 0 \leq D_i^t \leq W \cdot d_i, \tag{18b}$$

$$\kappa_i^t, \gamma_i^t, \xi_i^t \in \{0,1\}\ \forall i, t \in \mathcal{T}. \tag{18c}$$

In summary, the complete mixed-integer formulation reads

$$\max \sum_{i \in E_{out}} \sum_{t \in \mathcal{T}} f_i^t \cdot \Delta t \tag{19}$$

s. t.

(9), (10), (11), (13), (16), (17), (18)

To conclude this section, some remarks are in order.

Remark 4:

- Due to the nature of real-world problems, it is reasonable to optionally include an additional restriction on the worker distribution:

$$p_i^t = W \cdot \beta_i^t, \quad p_i^t \in \mathbb{Z}_0^+, \quad \forall i, t \in \mathcal{T}. \tag{20}$$

- The optimization problem consists of eight sets of different continuous variables $(c, u, \beta, f, g, h, A, D)$ and three sets of binaries (κ, γ, ξ). All variables depend on the number of edges and time steps. Hence, the problem size is $\mathcal{O}(n_t \cdot |E|)$, which restricts to work with relatively coarse time grids, compare Section 4.

3. Steady state analysis

A typical question arising in the context of ODEs is to study the long term behaviour of the system. In this section, we are interested in steady state solutions of the repair model. Considering time independent capacities, buffer levels and flows, we can simplify and reformulate the original model. Additionally, we can also optimize the flow distribution at branching nodes in order to increase the throughput inside the network and thus the outflow, cf. routing problems in [8, 10, 11, 14]. This leads to a so-called maximum flow problem, a well known problem in graph theory, see [5, 12, 20] for an overview.

Let us start with a definition.

Definition 3.1: A solution of (8) is called steady state solution if $\partial_t u_i(t) = 0$ and $\partial_t c_i(t) = 0$ as well as $\beta_i(t)$ is unchanging in $t\ \forall i \in E$.

First of all, we compute the capacities c_i for the steady state solution. The capacities in equilibrium are computed by

$$\min\{\frac{\mu_i - c_i}{\epsilon}, W d_i \beta_i\} - \min\{\frac{c_i}{\epsilon}, \alpha_i\} = 0 \quad \forall i. \tag{21}$$

For simplicity, we drop the time index t whenever the context is clear.

At this stage we assume, that all parameters including β_i are given. Then, the

steady state c_i can be determined in the following way:

$$c_i = \begin{cases} \mu_i - \epsilon\alpha_i, \text{ if } \mu_i \geq 2\epsilon\alpha_i \wedge \beta_i \geq \frac{\alpha_i}{Wd_i} & \text{(Case 1.1)} \\ \epsilon W d_i \beta_i \quad \text{if } \text{—— } \text{"} \text{—— } \wedge \beta_i < \frac{\alpha_i}{Wd_i} & \text{(Case 1.2)} \\ \frac{1}{2}\mu_i, \quad \text{if } \mu_i < 2\epsilon\alpha_i \wedge \beta_i \geq \frac{\mu_i}{2\epsilon W d_i} & \text{(Case 2.1)} \\ \epsilon W d_i \beta_i \quad \text{if } \text{—— } \text{"} \text{—— } \wedge \beta_i < \frac{\mu_i}{2\epsilon W d_i} & \text{(Case 2.2)} \end{cases} \tag{22}$$

It can be checked, that this choice of c_i really fulfills (21) by considering every case separately. Considering the limit process $\epsilon \to 0$, we get: Either $c_i = \mu_i$, if there are enough workers to balance the breakdown rate (Case 1.1), or $c_i = 0$ (Case 1.2). Case 2.1 and Case 2.2 will never occur, since μ_i is always positive.

Next, we investigate the buffer levels u_i in the ODE-constraint (3). In steady state, we have

$$[B \cdot f]_i + f_{ext,i} - f_i = 0, \quad \forall i. \tag{23}$$

Since c_i as well as u_i should be constant in steady state, $f_i := \min\{c_i, \frac{u_i}{\tau_i}\}$ is also constant. This means, if we find variables f_i, such that

$$[B \cdot f]_i + f_{ext,i} - f_i = 0 \quad \text{and} \tag{24}$$

$$0 \leq f_i \leq c_i \tag{25}$$

are met, we can set $u_i := f_i \cdot \tau_i$. In that way $f_i = \frac{u_i}{\tau_i} \leq c_i$ holds, and thus (3b) is automatically fulfilled. Apparently, equation (24) is only true, if the external inflow f_{ext} is constant as well.

3.1. *Steady state optimization problem*

In the original optimization problem (8), the goal is to find a worker distribution such that the outflow of the network is maximal. The same can be done in the stationary case. However, the optimization procedure is only interesting, if we do not previously fix the external inflow, but leave it variable, since

$$\sum_{i \in E} f_{ext,i} = \sum_{i \in E_{out}} f_i, \tag{26}$$

holds in steady state. We slighty reformulate (24) and end up with the following constraints:

$$[B \cdot f]_i - f_i + f_{ext,i} = 0 \qquad \forall i \in E_{in} \tag{27a}$$

$$[B \cdot f]_i - f_i = 0 \qquad \forall i \notin E_{in}. \tag{27b}$$

Consequently, the steady state optimization problem reads:

$$\max \sum_{i \in E_{out}} f_i \tag{28}$$

s. t.

(7), (8e), (22), (25), (27)

In order to solve (28) with respect to the worker distribution β and the external inflow f_{ext}, we linearize (22) and obtain again a linear mixed integer optimization problem $\forall i \in E$ (if not declared otherwise):

$$\max \sum_{i \in E_{out}} f_i \tag{29a}$$

s. t.

$$[B \cdot f]_i - f_i + f_{ext,i} = 0 \qquad \forall i \in E_{in} \tag{29b}$$

$$[B \cdot f]_i - f_i = 0 \qquad \forall i \notin E_{in} \tag{29c}$$

$$-d_i W(1-\delta_i) \leq \alpha_i - W d_i \beta_i \leq \alpha_i \delta_i \tag{29d}$$

$$-\mu_i \delta_i \leq c_i - \mu_i + \epsilon \alpha_i \leq (\mu_i + \epsilon \alpha_i) \cdot \delta_i \tag{29e}$$

$$-\epsilon W d_i (1-\delta_i) \leq c_i - \epsilon W d_i \beta_i \leq \mu_i (1-\delta_i) \tag{29f}$$

$$\sum_{i \in E} \beta_i = 1 \tag{29g}$$

$$0 \leq \beta_i \leq 1, \quad 0 \leq f_i \leq c_i, \quad 0 \leq c_i \leq \mu_i \tag{29h}$$

$$\delta_i \in \{0,1\}. \tag{29i}$$

Constraints (29d) to (29f) ensure, that c is set according to (22). (29d) has the effect, that δ_i is set to 0, when $\beta_i \geq \frac{\alpha_i}{W d_i}$, otherwise it is set to one. (29e) guarantees, that c_i is set to $\mu_i - \epsilon \alpha_i$, when $\delta_i = 0$. In the same way (29f) ensures, that c_i is set to $\epsilon W d_i \beta_i$ in the case $\delta_i = 1$.

3.2. *Connection to max flow problems*

At this point, it is rather simple to include an additional optimization task, namely the optimization of the flow distribution at branching nodes (nodes with more than one outgoing edge). As we have seen, the matrix B strictly prescribes the behaviour of the flow at vertices. However, we could instead use an incidence matrix, that only describes the incoming and outgoing edges of a vertex, without fixing the distribution rates. This makes the model more flexible and leads to larger flows as numerically shown later in Section 4.

The incidence matrix K is constructed as follows. Given a network with n edges and m vertices, we have $K \in \mathbb{Z}^{m \times n}$, whose elements are set in the following way:

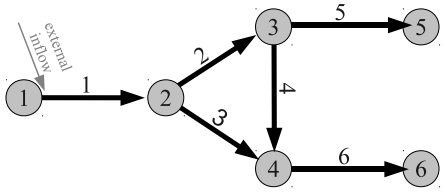
$$k_{v,i} = \begin{cases} 1, \text{ if } i \text{ is incoming edge of } v \\ -1, \text{ if } i \text{ is outgoing edge of } v \\ 0, \text{ else.} \end{cases}$$

A network with corresponding incidence matrix K is shown in Figure 3.

Consequently, we can include the issue of optimizing the flow distribution by exchanging constraint (27) with

$$[K \cdot f]_v = 0 \qquad \forall v \in V \backslash (V_{in} \cup V_{out}) \tag{30a}$$

$$[K \cdot f]_v + f_{ext,v} = 0 \qquad \forall i \in V_{in}. \tag{30b}$$



$$\longrightarrow \qquad K = \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$V_{in} = \{v_1\}$ and $V_{out} = \{v_5, v_6\}$.

Figure 3. Example of a network with corresponding matrix **K**

Hence, an improved steady state optimization problem is

$$\max \sum_{i \in E_{out}} f_i \tag{31}$$

s. t.

(7), (8e), (22), (25), (30)

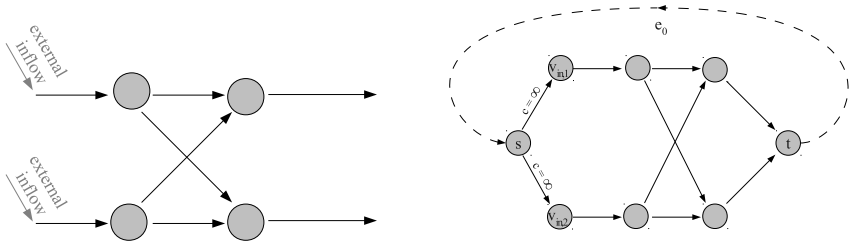
Another difference to the previous model (30) is, that the external inflow is specified at vertices, and not at edges. Since we usually want the inflow to enter the network at edges without predecessors, we can easily assign the external inflow to the startvertices without changing the setting.

Remark 1: It is also possible to use the incidence matrix K for the dynamic model (19). But this might lead to highly fluctuating flow distributions in the optimal solution. Since the rates do not explicitly appear as variables in the MIP, they can not be restricted to be constant in time, cf. [8].

Choosing a worker distribution β and computing the capacities c according to (22), we end up with a well known problem of graph theory, the Maximum Flow Problem (MFP). In the sequel, we will use theoretical results from MFPs to proof the existence of a solution to (31).

Lemma 3.2: *Given a network $G = (V, E)$ with properties $\alpha_i, d_i > 0$ and $\mu_i, \epsilon \geq 0$, $\forall i \in E$, there exists a feasible solution of (31) with $\sum_{i \in E_{out}} f_i = \sum_{v \in V_{in}} f_{ext,v} \geq 0$.*

Proof: Let β be an arbitrary worker distribution, satisfying (7). The upper bound of the flow is given by (22) and satisfies $c_i \geq 0$, $\forall i \in E$. The network can be transformed in the following way: We can imagine the external inflow as edges from a source vertex s to the point where the external inflow is supposed to enter the network. The upper bound c of these edges is set to infinity. In the same way, we can add an extra sink vertex t where all outflow edges are led to. Furthermore, we add an artificial edge e_0 from the sink to the source node, which represents the total flow-through, cf. Figure 4.
From (25) we know, that the lower bound of the flow in each edge is 0. This setting fulfills the conditions for the existence of a feasible flow circulation stated in Hoffman's circulation theorem, see [5], Theorem 3.8.2. Taking the properties of a flow for vertex s and t into account as defined in graph theory, it directly follows that $f_{e_0} = \sum_{i \in E_{out}} f_i = \sum_{v \in V_{in}} f_{ext,v}$ holds, due to the construction of the transformed network. □



(a) (b)

Figure 4. Transformed network

4. Computational experiments

For computational experiments we use two different approaches. For the first small test case discussed in subsection 4.1, we implemented the optimization problem (8) in Matlab 7.5 using the function *fmincon*, which is a solver for nonlinear optimization problems, see [18]. This approach works quite well as long as the test cases are small. The disadvantage of this solver is, that it often gets stuck in local optima and it does not allow to restrict the worker distribution to integer workers. For these reasons we derived the mixed integer formulation (19) which can be used by cplex 12.1.0, a commercial solver for linear mixed integer problems developed by IBM, formerly Ilog, see [16]. It uses a branch and cut algorithm providing the user with currently found primal as well as dual bounds during the optimization process. In the case, that the duality gap tends to zero, the user can be sure, that the provided solution is indeed globally optimal. Furthermore, this method has the advantage, that we can easily restrict the workers to integer numbers, which is indeed meaningful for real world applications. In subsection 4.2 this method is applied to a branched network where also steady state studies are carried out. All computations are performed on a PC equipped with 16GB Ram, Intel(R) Xeon(R) CPU 5160 @ 3.00GHz.

4.1. *Serial processors*

Initially, we sketch the impact of the numerical parameters on the result as well as introduce the modelling aspect of worker changes during the time horizon. Therefore, we take a small test example. We consider two machines in a row with a fixed parameter setting, see Figure 5 and 6.

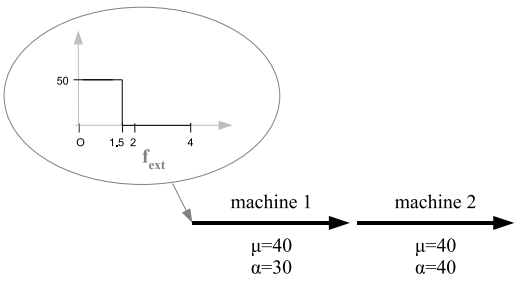


Figure 5. Two serial processors

time horizon:	$T = 4$
throughput time:	$\tau = 0.25$
workers:	$W = 40$
Total inflow: $\int_0^T f_{ext}(\tilde{t})d\tilde{t} = \sum_{t\in\mathcal{T}} f_{ext}^t \Delta t$	=75 parts

Figure 6. Parameter setting

The time horizon is $T = 4$, the throughput times at buffers are $\tau_i = 0.25$ for every machine and 40 workers are available. The external inflow enters the network at the first machine. In the first 1.5 time units, we have an inflow of 50 and thus

$\int_0^T f_{ext}(\tilde{t})d\tilde{t} = 75$ parts are introduced into the system. Furthermore, here and in all following examples, the repair times d_i are set to 1 for all edges.

As initial condition, we set c_0 to full capacities and assume empty queues in the beginning (i.e. $u_{0i} = 0$). Furthermore, we provide a start solution, where the workers are equally distributed among the edges, i.e. we have 20 workers at each machine.

4.1.1. Numerical investigations

We perform a simulation assuming the worker distribution rate β to be constant for the whole time horizon. We compute the objective function value (8a) for different values of β, using the Matlab routine *fmincon* with explicit Euler discritization for the ODE-constraints. We let β_1 go from 0 to 1 in steps of length 0.001. In Figure 7, we compare the simulation results operforf different time grids. We can observe that the optimal objective function value tends to the same value, even for coarse time grids.

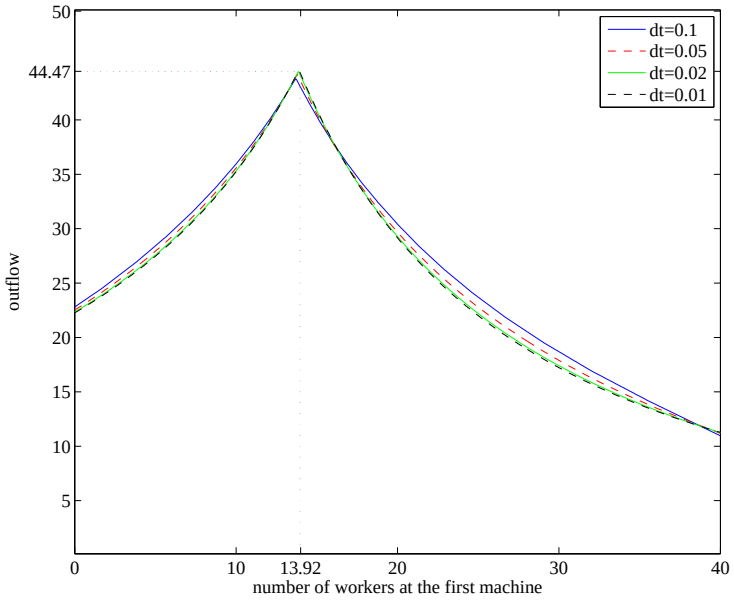


Figure 7. Comparison of simulation using different time grid sizes.

In this setting, the maximal outflow of 44.47 units is achieved, if 13.92 workers are sent to the first machine and 26.08 to the second one. As we can see in Table 1 and in accordance with Figure 7, the conservation of mass is kept with an accuracy that depends on the time grid size.

Table 1. verifying conservation of mass,

Δt	opt. worker distr.	max outflow	final queues	$\sum$
0.1	[13.716, 26.284]	43.756	31.457	75.213
0.05	[13.803, 26.197]	44.163	30.888	75.050
0.02	[13.871, 26.129]	44.453	30.558	75.011
0.01	[13.921, 26.079]	44.471	30.532	75.003

The total inflow is $\sum_{t\in\mathcal{T}} f_{ext}^t \Delta t = 75$.

4.1.2. Worker changes

In a next step, we illustrate the modeling aspect of worker changes. We have the option, to vary the worker schedule at certain points within the time horizon. We allow the workers to change their position once in the middle of the time horizon. We fix the time grid size to $\Delta t = 0.01$ and simulate the objective function value varying β_1^t from 0 to 1 with step width 0.001 (β_2 of the second machine automatically varies since $\beta_2^t = 1 - \beta_1^t$). Since β_1^t has two values (one for each time period), we end up with a 3d-plot showing the objective function value for all combinations of $\beta_1^t, t \in [0; 2]$ and $\beta_1^t, t \in (2, 4]$. The result is depicted in Figure 8.

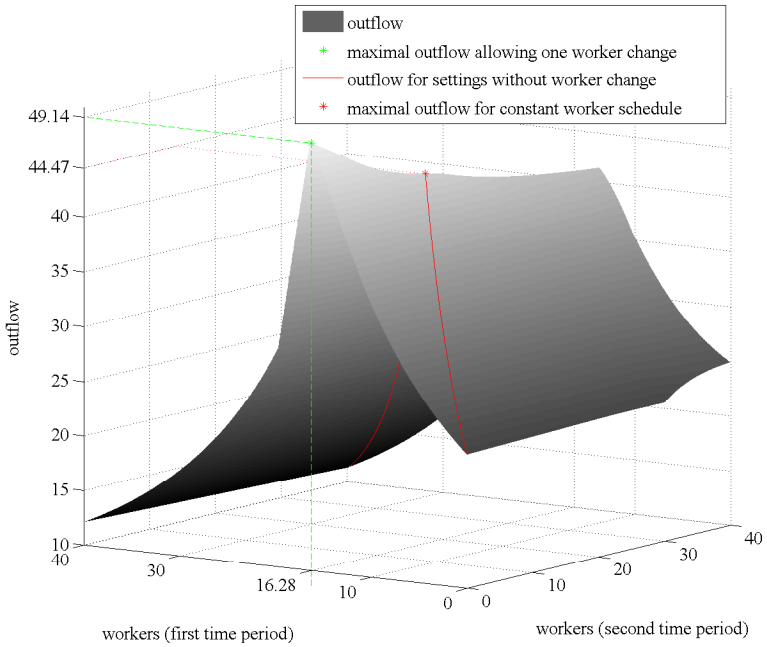
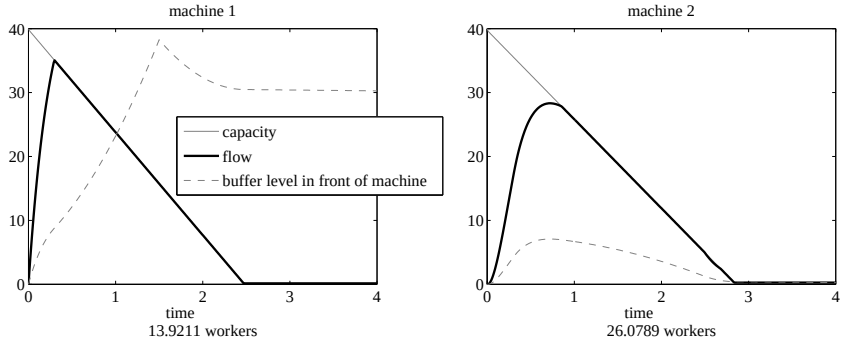


Figure 8. Outflow depending on the number of workers at the first machine, including one worker change in the middle of the time horizon.

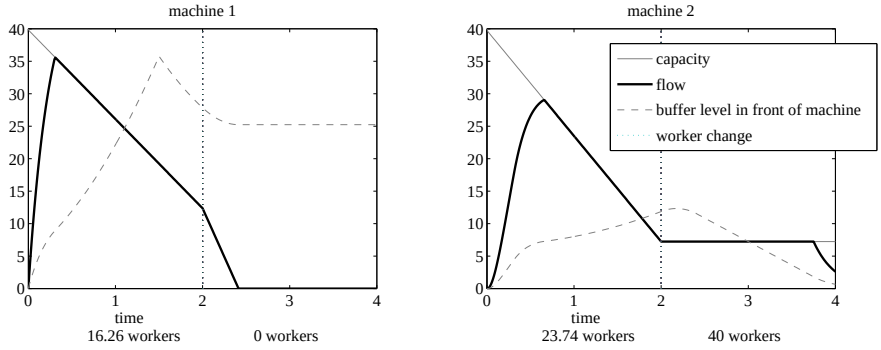
Obviously, at a first result, allowing one worker change within the time horizon leads to an improvement of the optimal solution (49.14 units compared to 44.47 units). To understand this behavior, we shall have a closer look at the evolution of flow, capacities and buffer levels as well, which are plotted in Figure 9(a).

In all these plots, we observe that the number of workers at a machine influence the slope of the capacity evolution. Unless the capacity is neither 0 nor has reached its maximal level μ, it can be described by a (piecewise) straight line with slope $W\beta_i d_i - \alpha_i$ (cf. equation (6)). Since the breakdown rate of the second machine is 40, we need all 40 workers to keep the capacity at the same level. This happens in the time period after the workers have changed, see Figure 9(b). In this way, the flow is sustained leading to a larger total outflow value compared to a fixed worker schedule.

Remark 1: It is not always the case, that a unique maximum is reached. On the contrary, in more complicated settings many local maxima may occur. In such cases the *fmincon* solver is not reliable anymore, since it often gets stuck in local optima. Another drawback of *fmincon* is, that we cannot stick to schedules with integer workers.



(a) Constant worker schedule



(b) One worker change in the middle of the time horizon

Figure 9. Optimal solution for serial processors

4.2. *Branched networks*

So far, the serial processor test case is a nice example to get insight and feeling for the dynamics involved in the repair worker assignment model. After this numerical experiments in Matlab, we now have a different focus. First of all, we analyse the steady state problem (29) and point out, in which way the obtained information can be exploited for the dynamic model (19). The models, formulated as linear mixed integer programming problems, are solved by CPLEX [16]. We extend our studies to a more general network with 12 edges, as shown in Figure 10 and restrict the worker distributions to integer values only due to the easier applicability to real world problems. We allow external inflow for the first two edges and are interested in maximising the outflow at edges 11 and 12.

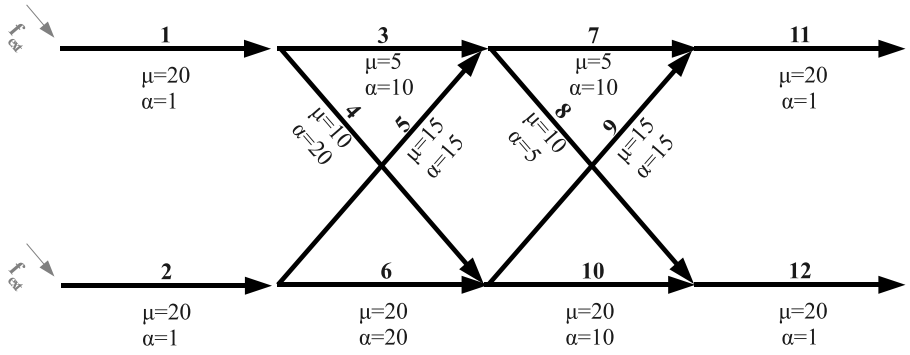


Figure 10. Branched network with 12 edges where $d_i = 1$ for all machines.

4.2.1. Steady state studies

Before we prescribe the external inflow and compute the optimal solution of the dynamic model (19), we first have a deeper look at the steady state solutions, described in Section 3. Different from the dynamic model, the external inflow of the steady state model (29) is not given a priori, but is maximized simulatenously with the outflow.

Analysis on the amount of available repair workers

From case 1.1 of equation (22) with $\epsilon = 0$ we can deduce, that we need at least $\frac{\alpha_i}{d_i}$ workers in order to keep the capacity of machine i to its maximal level. In our setting d_i is set to one for all i and the breakdown rates α sum up to 109. This means, that at least 109 workers are necessary avoid capacity drops. Since employing workers is expensive, it is rewarding to check, how we can cope with less manpower.

The question arises, how many workers we would at least need to get a steady state flow-through greater than zero. In the case that we do not previously fix the flow distribution at the nodes as explained in subsection 3.2, we can find the answer in the following way: Assume, that the maximal capacity μ_i is greater than zero for all machines. As explained before, the capacity of a machine can only be sustained, if at least $\frac{\alpha_i}{d_i}$ workes are allocated to it. We can find the least manpower consuming path through the network by using a standard shortest path algorithm such as Djikstra algorithm, for more details see [12] and [20] amongst others. The flow-through of this path is bounded by its bottleneck, which is the machine with the smallest capacity. For our testcase, we need at least 17 workers contributed along the shortest path to get a steady state solution greater than zero. In this case, the flow-through is 5 parts per time unit, see Figure 11.

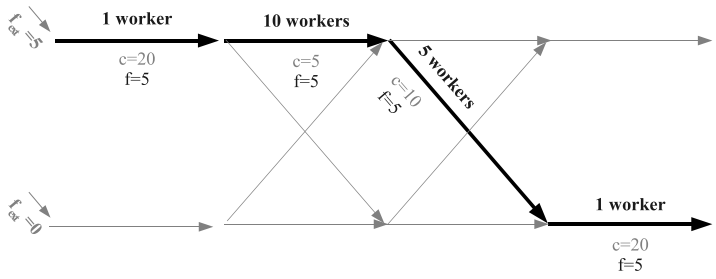


Figure 11. The least manpower consuming steady state solution greater than zero. The resulting flow-through is 5 parts per unit time requiring a minimum of 17 workers.

When we previously fix the distribution behaviour of the flow as in Subsection 3.1, for example to equal distribution between the suceeding edges, we need a lot more workers to get a positive flow-through greater. This is due to the fact, that once an edge transmits a flow, all its succeeding edges must also have a capacity greater than zero, such that the flow can be distributed in the prescribed way. For our testcase, we need at least 73 workers to get a positive steady state flow. The resulting flow-through is 10 parts per unit time. For details, see Figure 12.

Moreover, it is interesting to compute the maximal steady state solution, when we have no capacity drop. If the flow distribution at branching nodes is not previously fixed, we can find the solution via the Ford-Fulkerson-Algorithm [5] using the maximal capacities $c_i = \mu_i$ as upper bounds. The result is shown in Figure 13.

This gives us an upper bound for the maximal flow-through. In our case, it is 35 parts per time unit.

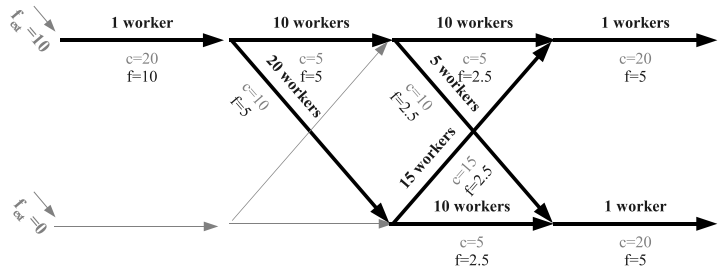


Figure 12. The least manpower consuming steady state solution greater than zero, for equally distributed flow at branching nodes. The resulting flow-through is 10 parts per unit time and the necessary number of workers is $W = 73$.

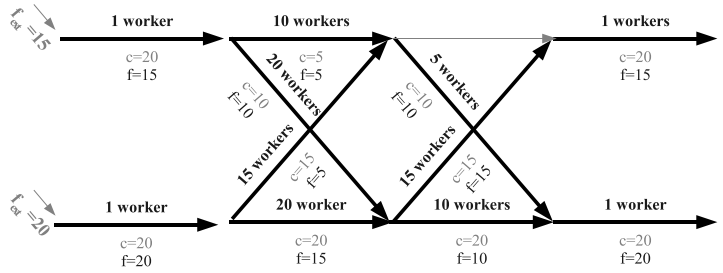


Figure 13. The maximal static flow-through when all capacities are at their maximal level.

Under-staffed settings

From the above analysis we know that finding the optimal worker distribution is only interesting in the case, that we have less than 109 workers available. Otherwise, we can always distribute the workers in a way, that no capacity loss occurs.

In the following, we consider two scenarios, where the total number of workers is set to 30 ($\rightarrow$ highly under-staffed) and to 100 ($\rightarrow$ slightly under-staffed) respectively.

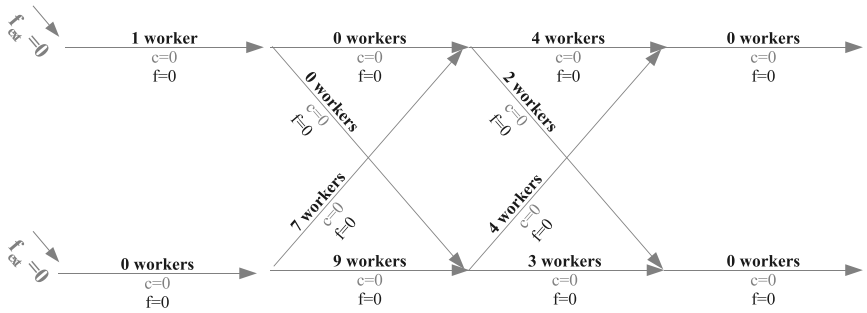
Moreover, we compare both versions of the steady state optimization problem (29): First we use matrix B and therewith a fixed flow distribution at branching nodes, and for the second run we exchange B with the incidence matrix K, see (30), leading to variable flow distributions that are subject to the optimization process.

The resulting maximal flow-through of the different settings is depicted in Figure 14 for 30 workers and in Figure 15 for 100 workers.

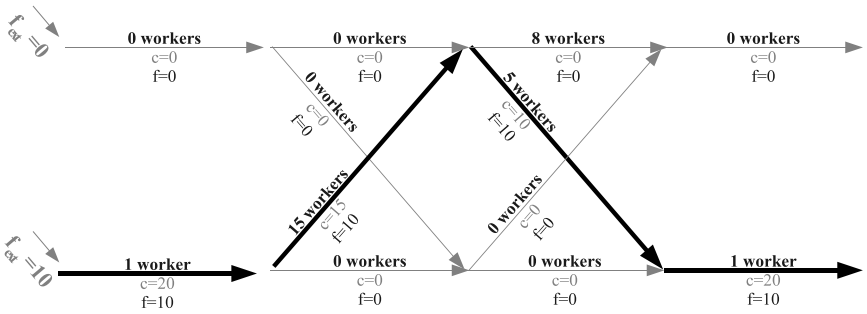
In the highly under-staffed scenario with fixed flow distribution (see Subfigure 14(a)), it is not possible to allocate the workers in a way to obtain positive solution. All machines are out of order and no flow is able to go through. However, if we leave the distribution of flow up to optimization, we can find a solution, where a flow-through of 10 parts per time unit can be provided, on the only functioning path through the network (see Subfigure 14(b)).

As expected, we get a much better solution, when we increase the number of workers to 100 (see Figure 15). Now, the setting with fixed flow distribution allows a maximal flow-through of 20 (see Subfigure 15(a)), whereas the additional optimization of the flow distribution increases the flow-through to 35 (see Subfigure 15(b)). As shown above, this is already the upper bound of steady flow-through with respect to the number of repair workers.

The steady state solutions can be useful for the dynamic model. As explained in the next subsection, the steady state analysis provides us with a qualitatively good

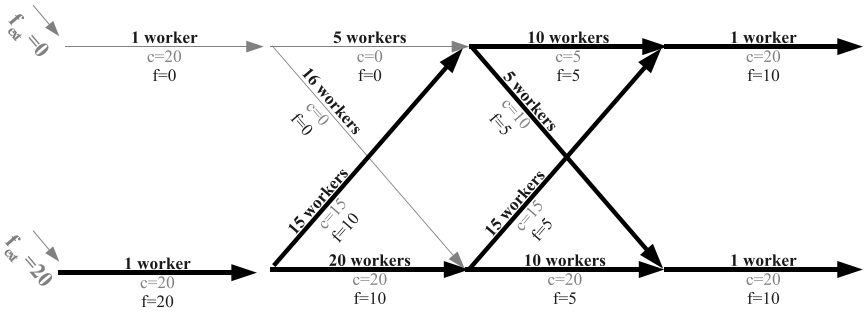


(a) Fixed flow distribution

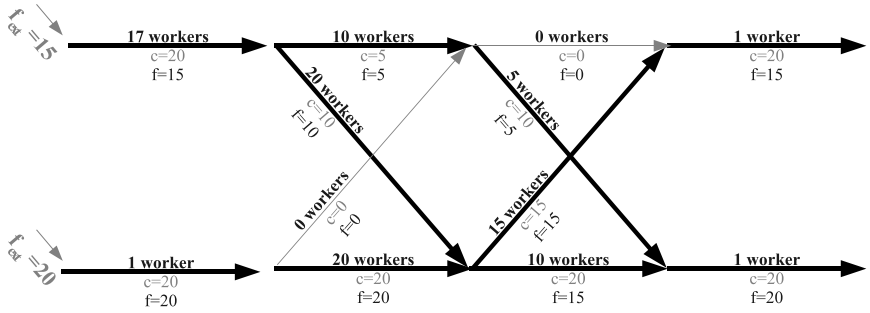


(b) Optimized flow distribution

Figure 14. Maximal flow-through, scenario with 30 workers



(a) Fixed flow distribution



(b) Optimized flow distribution

Figure 15. Maximal flow-through, scenario with 100 workers

start solution for the dynamic MIP (19), leading to significant runtime reductions of the optimization procedure. Furthermore, we can observe, that optimization of the flow distribution at branching nodes leads to a considerable gain of outflow. This does not only hold for the steady state case but also for the dynamic setting, as described in the sequel.

4.2.2. Dynamic repair model

Now, we move on to the dynamic repair model (19). As before, we use the flow distribution matrix B that divides the flow in equal shares among the suceeding edges at branching nodes.

Different to the steady state model, we have to fix the external inflow function in the dynamic setting. We choose $f_{ext} \equiv 20$ for edge 1 as well as for edge 2. The timehorizon T is set to 5 and the timegridsize to $\Delta t = 0.1$. As initial conditions, the network is empty, i.e. buffers and flows are equal to zero for $t = 0$ and the capacities are set to its maximal values $c_i^0 = \mu_i$. As in the previous subsection, we again consider the highly under-staffed setting with 30 workers as well as the slightly under-staffed one where 100 repair workers are available.

Due to the high complexity of the dynamic problem (19) it is advisable to provide a start solution in order to speed up computation time. A feasible start solution can easily be computed by fixing the worker assignment for all machines and computing the forward solutions for the capacity and buffer conditions according to (9b) and (9a). The optimal worker distribution of the corresponding steady state problem, see Figures 14(a) and 15(a), turns out to be an adequate start solution. In Table 2, this procedure is compared to the use of another start solution, where workers are equally distributed amongst the machines. The initial objective function value provided by the steady state solution as well as the optimization time is significantly better.

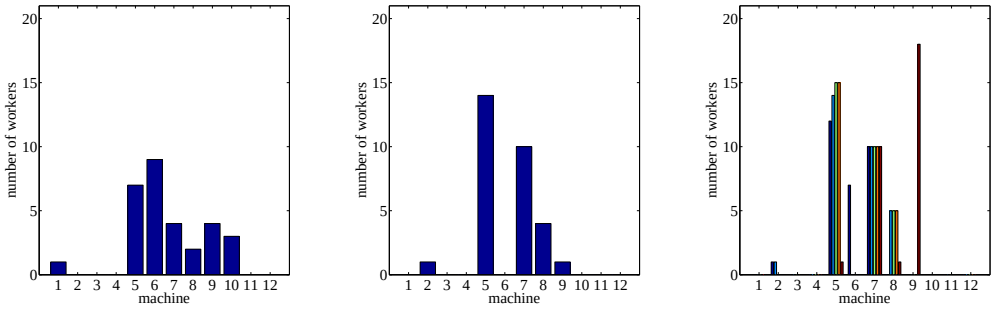
Table 2. Optimization results for the dynamic repair problem using different start solutions.

#workers	start solution	optimization time	outflow of start sol.	optimal outflow
30	equally distr. workers	504.55 s	10.27	41.88
	steady state solution	316.08 s	16.52	—"—
changes allowed	steady state solution	> 3 days	16.52	42.73
100	equally distr. workers	1794.47 s	28.14	120.49
	steady state solution	569.53 s	90.91	—"—
changes allowed	steady state solution	154.74 s	90.91	121.12

Additionally we can allow the workers to change position after each time unit. For this option we use again the steady state solution as a starting point for optimization. The results are listed in the 3rd and last row of Table 2. The obtained outflow is slightly higher in that setting. However, the complexity of the problem increases this way. As far as computation time is concerned, it takes the optimization algorithm several days to find the optimal solution of the highly under-staffed setting. Though, if 100 workers are available, optimization time, namely 154.74 seconds, is unexpectedly short when worker changes are allowed. This can be explained by comparing the optimal worker distribution with the start solution, which is shown in the following figures that are explained in the sequel.

In Figure 16 and 17, the worker distributions are plotted for the highly and slightly under-staffed setting respectively. The left plot shows the optimal worker distribution of the steady state case and the next one represents the optimal solution of the dynamic model. The rightmost plot depicts the optimal distribution when workers are allowed to change position after each time unit. Here, each bar

is devided into 5 supbars that represent the number of workers being assigned to the respective machine at each of the 5 time units. In Figure 17 we can see, how close the start solution (Figure 17(a)) already is to the optimum (Figure 17(c)). This fact explains the short optimization time.

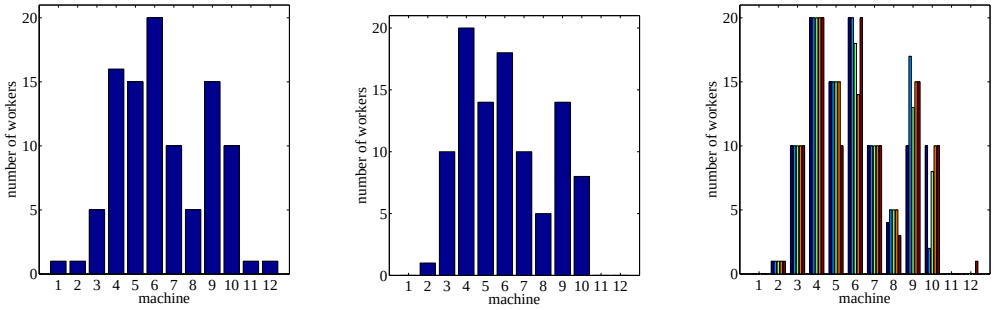


(a) Optimal worker distribution of the steady state case

(b) Optimal solution for the dynamic model

(c) Optimal solution for the dynamic model allowing the workers to change position after each time unit

Figure 16. Optimal worker distribution for the highly under-staffed setting, i.e. $W = 30$.



(a) Optimal worker distribution of the steady state case

(b) Optimal solution for the dynamic model

(c) Optimal solution for the dynamic model allowing the workers to change position after each time unit

Figure 17. Optimal worker distribution for the slightly under-staffed setting, i.e. $W = 100$.

Changing the flow distribution at branching nodes

In the sequel, we will use an important observation concerning the previously described steady state analysis. Remember, that the steady flow-through can be significantly improved, when the flow distribution at branching nodes is not a priori fixed. A straightforward idea would be to include this flexibility as well into the dynamic model (19) by exchanging the flow distribution matrix B by the incidence matrix K analogously as done for the steady case in Subsection 3.2. However, this ansatz incloses a significant drawback. Since the distribution rates of the flow do not appear explicitely as parameters in the formulation of the problem when the incidence matrix is used, it is not possible to restrict to constant distribution rates. Consequently, we can not avoid the undesired effect, that solutions contain highly fluctuating flow distributions. For that reason, we prefer to track another idea. We use the optimized flow distribution of the steady state case for the dynamic model (19) by adapting matrix B accordingly.

step 1: Compute the steady state solution with variable flow distribution (30).
step 2: Construct the flow distribution matrix B according to the distribution of the steady state obtained in step 1.
step 3: Solve the dynamic repair model (19) using B and taking the optimal worker distribution of step 1 as start solution.

In table 3 the corresponding optimization results are listed.

Table 3. Optimization results for the dynamic repair problem using optimal flow distribution rates of the steady state analysis.

# workers	changes allowed?	optimization time	opti. gap	outflow of. start sol.	optimal outflow	improvement to previous flow distr.
30	no	3207.46 s	0 %	42.56	45.05	7.56 %
	yes	$>$ 3 days	2.33 %	42.56	$\in [49.56, 50.71]$	$>$ 15.98 %
100	no	0.84 s	0 %	126.28	126.28	4.80 %
	yes	0.84 s	0 %	126.28	126.28	4.26 %

The last column shows the considerable gain of outflow by using the optimized matrix B instead of equal flow distribution, see Table 2. Furthermore, it is interesting to have a look at the computation time. For the highly under-staffed setting optimization takes notedly longer. When allowing worker changes, the optimality gap of the algorithm could not even be completely closed after three days. On the other hand, the gain of outflow is noteworthy, especially when workers are allowed to change their position after each time unit. When 100 workers are available, the optimal steady state solution turns out to be already optimal for the dynamic model, even for the case in which we allow worker changes. Hence, the optimization time is with 0.84 seconds extremely short.

The solutions of the different settings are illustrated in Figure 18.

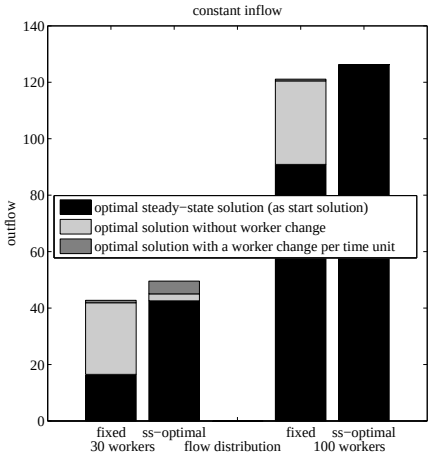


Figure 18. Comparison of outflow of different settings

It is remarkable, that the optimal steady state worker distribution is already really close to the optimal solution of the dynamic model in the case, that we use the flow distribution that is optimal for the steady state case (in the figure denoted by "ss-optimal").

Summarising the numerical observations, we can underline the benefit of the steady state analysis. Note, that the steady state problem (29) is much faster solvable than the far more complex dynamic MIP (19), where we need the whole set of variables for each single time step. First of all, the steady state analysis provides us with a qualitatively good starting solution that leads to significant runtime

reductions for the optimization of the dynamic model. Secondly, the additional optimization of the flow distribution in the steady state case, endows us with valuable information how to increase the outflow of the dynamic model, given that the flow distribution of the corresponding application is adaptable accordingly.

Conclusions

We have presented a novel production model for the time-changing repair worker assignment. The main idea is to keep the system performance optimal whenever machines have failed and must be repaired. In general, available workers are limited and therefore a decision has to be made which machines are repaired first. The resulting optimization problem has been intensively analysed and numerical case studies comparing fixed and time-changing schedules have been performed. However, we could answer many questions there are still numerous open topics for future considerations. Besides the fact that there is great need for faster solution procedures, it is interesting to think about the influence of stochastic breakdown rates α instead of using experience values. Intuitively, this will lead to a completely different setting. But also the question of an optimal schedule in the sense that we seek flexible points in time at which the workers must move to broken machines is an attractive issue. This is not clear so far and will be of our interest in future research projects.

Acknowledgements

This work was financially supported by the DAAD research grants no. D/08/11076, 50727872, 50021880 and the DFG projects HE5386/6-1 and HE5386/8-1.

References

[1] D. Armbruster, C. de Beer, M. Freitag, T. Jagalski, and C. Ringhofer, *Autonomous Control of Production Networks using a Pheromone Approach*, Physica A, Vol. 363(1), pp. 104 – 114, 2006.
[2] D. Armbruster, P. Degond, and C. Ringhofer, *A Model for the Dynamics of Large Queuing Networks and Supply Chains*, SIAM J. on Applied Mathematics, Vol. 66(3), pp. 896 – 920, 2006.
[3] C. D'Apice and R. Manzo, *A Fluid Dynamic Model for Supply Chains*, Netw. Heterog. Media, Vol. 1(3), pp. 379 – 398, 2006.
[4] C. D'Apice, R. Manzo, and B. Piccoli, *Modelling supply networks with partial differential equations*, Quart. Appl. Math., Vol. 362, pp. 374 – 386, 2010.
[5] J. Bang-Jensen and G. Gutin, *Digraphs: Theory, Algorithms and Applications*, Springer-Verlag, London, 2001.
[6] P. Brucker, *Scheduling Algorithms*, Springer-Verlag, New York, Berlin, Heidelberg, 2004.
[7] C. F. Daganzo, *A Theory of Supply Chains*, Springer-Verlag, New York, Berlin, Heidelberg, 2003.
[8] A. Fügenschuh, S. Göttlich, M. Herty, A. Klar, and A. Martin, *A Discrete Optimization Approach to Large Scale Supply Networks Based on Partial Differential Equations*, SIAM J. on Scientific Computing, Vol. 30(3), pp. 1490 – 1507, 2008.
[9] S. Göttlich, M. Herty, and A. Klar, *Network Models for Supply Chains*, Comm. Math. Sci., Vol. 3(4), pp. 545 – 559, 2005.
[10] S. Göttlich, M. Herty and A. Klar, *Modelling and Optimization of Supply Chains on Complex Networks*, Comm. Math. Sci., Vol. 4(2), pp. 315 – 330, 2006.
[11] S. Göttlich, M. Herty, and C. Ringhofer, *Optimization of order policies in supply networks*, European J. of Operations Research, Vol. 202(2), pp. 456-465, 2010.
[12] H. W. Hamacher and K. Klamroth, *Linear and Network Optimization*, Vieweg+Teubner-Verlag, 2006.
[13] D. Helbing, *Production, supply, and traffic systems: a unified description*, Hoogendoorn, S. P. et al. (eds.), Traffic and granular flow 03, Springer-Verlag, Berlin, pp. 173 – 188, 2005.
[14] M. Herty and C. Ringhofer, *Optimization For Supply Chain Models With Policies*, Phys. A, Vol. 380, pp. 651 – 664, 2007.
[15] K. Hoffman and M. Padberg, *Solving airline crew-scheduling problem by branch-and-cut*, Management Science, Vol. 39, pp. 657 682, 1993.
[16] IBM ILOG CPLEX, IBM Deutschland GmbH 71137 Ehningen. Information available at URL http://www.ibm.com/software/products/de/de/ilogcple/.

[17] J. Kallrath, *Gemischt-ganzzahlige Optimierung: Modellierung in der Praxis,* Vieweg-Verlag, Wiesbaden, 2002.
[18] Matlab Version R2010a, MathWorks, Inc., 3 Apple Hill Drive, Natick, MA. Information available at URL http://www.mathworks.com/.
[19] K. Neumann, C. Schwindt and J. Zimmermann, *Project Scheduling with Time Windows and Scarce Resources*, Springer-Verlag, New York, Berlin, Heidelberg, 2002.
[20] S. O. Krumke and H. Noltemeier, *Graphentheoretische Konzepte und Algorithmen*, Vieweg+Teubner-Verlag, 2006.