COMMUN. MATH. SCI.
Vol. 14, No. 7, pp. 1799–1820

ROUTING STRATEGIES IN PRODUCTION NETWORKS WITH RANDOM BREAKDOWNS*

SIMONE GÖTTLICH† AND SEBASTIAN KÜHN‡

Abstract. Routing strategies in unreliable production networks are an essential tool to meet given demands and to avoid high inventory levels. Therefore, we are interested in studying state-independent and state-dependent control policies to maximize the total throughput of the production network. Different from M/M/1 queuing theory, the underlying model is based on partial and ordinary differential equations with random breakdowns capturing the time-varying behavior of the system. The key idea is to numerically compare suitable routing strategies with results computed by nonlinear optimization techniques. We comment on the efficiency of the proposed methods and their qualitative behavior as well.

Key words. Production networks, differential equations, random breakdowns, routing strategies, optimal control.

AMS subject classifications. 90B15, 65Mxx, 90C30.

1. Introduction

Continuous models for the modeling, simulation and optimization of production networks has become an important research field during the last decades. In contrast to widely used models based on discrete optimization approaches [30, 33], discrete event simulations [3, 26] or queuing theory [4, 8], continuous models allow for a detailed time-dependent description of the production process using quantities such as the part density or the flow of goods [5, 10, 11, 12, 13].

Time-continuous network models of serial networks have been introduced in [2] for the first time. Therein, the authors rigorously derived a differential equation, namely a conservation law, for the part density from a discrete event simulation. In [19, 20], this model has been reformulated by installing buffer of infinite size in front of each individual processors. So far, these models have been considered mostly from the deterministic point of view, but it is possible to include stochastic effects in a straightforward way. For instance, under certain assumptions for the availability of processors, averaged densities can be computed either analytically [16] or numerically [22] using Monte-Carlo simulations. In both approaches, random breakdowns of processors are modeled as capacity drops at exponentially distributed points in time. We briefly describe the coupling of the stochastic process to the dynamics of the production system in Section 2.

For optimization purposes, the computation of the maximal throughput or the minimal buffer loads are of main interest. There exists a broad variety of literature related to this topic with a focus on the optimal routing of goods or cars [20, 24, 28], inflow optimization [14], or demand tracking [25]. However, the combination of continuous randomly perturbed production models and mathematical optimization issues has been

*Received: October 28, 2014; accepted (in revised form): June 10, 2015. Communicated by Pierre Degond.
This work was financially supported by the DAAD project "Transport network modeling and analysis" (Project-ID 57049018) and by the BMBF project KinOpt. Special thanks go to Stephan Martin and Thorsten Sickenberger for fruitful discussions and Markus Erbrich for his help in generating sample scenarios.

†University of Mannheim, Department of Mathematics, 68131 Mannheim, Germany (goettlich@math.uni-mannheim.de).

‡University of Mannheim, Department of Mathematics, 68131 Mannheim, Germany (kuehn@math.uni-mannheim.de).

less investigated heretofore. In other words, the challenge we face here is the optimal control of a nonlinear stochastic model relying on differential equations. That means we need to think about suitable optimization strategies and algorithms as well. We emphasize different solution approaches for the optimal routing problem, where the overall goal is to efficiently distribute goods through the system to achieve high throughputs. Major applications for the optimal routing problem are, for example, packet flow on data networks [7] or traffic flow on road networks [6, 9, 15, 23, 28].

In this work, our contribution will follow two central ideas. Due to the complexity of our modeling approach, a detailed analytical study of the routing problem is hardly possible. Therefore, we stick to a numerical study in our investigations and propose routing strategies (or policies) in a heuristic manner on the one hand and optimal solutions obtained by nonlinear optimization on the other hand. The control strategies may either depend on the current state of the system or not. In this way, we are able to include the time-varying behavior of the system more precisely. We still see in Section 4 how this additional information will influence the system optimum. Similar ideas can be found, for example, in queuing theory, where a variety of literature related to routing decisions exists; see [1, 27, 31, 32] for an overview. However, these techniques do not directly apply to our approach due to the fluctuations resulting from the random breakdowns of processors. Motivated by queuing theory, we develop problem-adapted routing strategies and approximate expectations of the system using a large number of Monte-Carlo runs. To assess the impact of the results achieved, we present an algorithm to solve the stochastic control problem directly. The latter can be interpreted as an optimization model restricted by differential equations. It is numerically solved using a rolling time horizon approach to really include all occurring random failures (cf. Section 3). This method is nonstandard and computationally very costly, and it often gets stuck at local approximations. To remedy this drawback, heuristic routing strategies offer an alternative and less expensive way to approximate or even reach a system optimum; see computational experiments in Section 4. From a numerical point of view, we try to find the most suitable strategy to reach high outputs and low buffer loads while also taking into account the network topology and different arrival rates.

2. Modeling of production networks and routing strategies

In this section, we briefly discuss a mathematical model to describe the flow of goods in production networks with random breakdowns of processors originally introduced in [22]. Here, breakdowns are modeled by a two-state process with exponentially distributed switching times between on and off states. We also present several routing strategies or policies to distribute the product flow through the system. We mainly distinguish between two types of strategies: state-independent and state-dependent policies.

2.1. Stochastic network model with random breakdowns. The modeling and numerical simulation of a stochastic time-dependent production model including random breakdowns is presented in [22]. In this work, this model is used to describe the fundamental dynamic behavior and is coupled to routing strategies or control policies, respectively.

To introduce the model, we first set a couple of notations. With $(\mathcal{V},\mathcal{A})$, we denote a directed graph consisting of a set of arcs $\mathcal{A}$ and a set of vertices $\mathcal{V}$ and define $N=|\mathcal{V}|$, $M=|\mathcal{A}|$. For any fixed vertex $v\in\mathcal{V}$, the set of ingoing arcs is denoted by δ_v^- and the set of outgoing arcs by δ_v^+; see Figure 2.3. Each processor is represented by an arc $e\in\mathcal{A}$ with an associated queue or buffer in front of it. We assume that each processor has a non-physical length, the so-called degree of completion described by the variable x.

The degree of completion is normed to the unit interval $[0,1]$, where $x=0$ indicates the entering and $x=1$ the exiting of parts. A vertex $v\in\mathcal{V}$ without any predecessor represents an inflow point to the production network. We denote the set of all these vertices by $\mathcal{V}_{\text{in}}=\{v\in\mathcal{V}\,|\,|\delta_v^-|=0\}$. The time-varying influx is externally given and denoted by $G_{\text{in}}^v(t)$ for all $v\in\mathcal{V}_{\text{in}}$. Accordingly, we define $\mathcal{V}_{\text{out}}=\{v\in\mathcal{V}\,|\,|\delta_v^+|=0\}$ as the set of all vertices where goods leave the production network. Furthermore, let $s:\mathcal{A}\to\mathcal{V}$ map an arc onto its vertex of origin.

The considered time span is $[0,T]$. We assume that processors may break down eventually and get restarted again within the time horizon T. Following [22], we define a two-state stochastic process

$$\begin{aligned} r^e:\mathbb{R}_{\geq 0}\times\Omega &\longrightarrow \{0,1\}\\ t\times\omega &\longmapsto r^e(t,\omega)\end{aligned} \tag{2.1}$$

for each processor $e\in\mathcal{A}$ indicating whether the processor is on, i.e. $r^e(t,\omega)=1$, or off, i.e. $r^e(t,\omega)=0$. Intermediate states are not possible; see Figure 2.1. Furthermore, we initialize the states of the processors by

$$r^e(0,\omega)=r_0^e, \tag{2.2}$$

where we usually choose all processors being on, i.e. $r_0^e=1$.

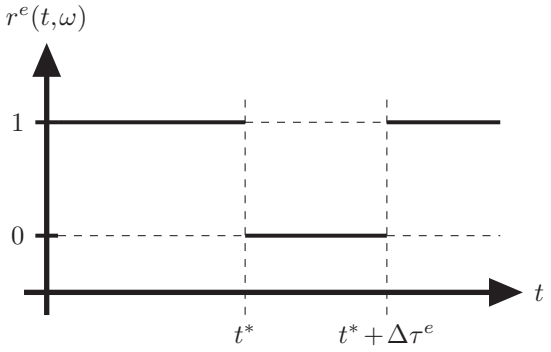

FIG. 2.1. *Realization of a two-state-process* (2.1).

The state process r^e depends on both the time t and the random sample $\omega\in\Omega$. Thus, for a fixed time $t\geq 0$, $r^e(t,\cdot)$ is a binary random variable, whereas, for a fixed random sample $\omega\in\Omega$, $r^e(\cdot,\omega)$ is a realization of the state process. We call a change in the state $r^e(t^*,\omega)$ of a processor $e\in\mathcal{A}$ at time t^* a *switching*. To model these, we assume that the switchings are independent of the queue load, the load of the processor, and the state of other processors. This allows us to introduce the *mean time between failures* (MTBF) τ_{on}^e and the *mean repair time* (MRT) τ_{off}^e for each processor e. The former describes the mean time for a switching from $r^e=1$ to $r^e=0$, while the latter defines the mean time for a processor being broken, i.e. $r^e=0$, before switching back to operating, i.e. $r^e=1$. Then, for each processor, the time $\Delta\tau^e$ between two switchings at t^* and $t^*+\Delta\tau^e$ is chosen randomly from the exponential distribution with density function $Exp(t;\lambda)$ and the rate parameter

$$\lambda=\lambda(r^e(t^*,\omega))=\begin{cases}1/\tau_{\text{on}}^e & \text{if } r^e(t^*,\omega)=1,\\ 1/\tau_{\text{off}}^e & \text{if } r^e(t^*,\omega)=0.\end{cases} \tag{2.3}$$

Having the modeling of breakdowns at hand, we can introduce the stochastic production network model as follows. We assume that each processor $e \in \mathcal{A}$ works with a constant velocity v^e and has a maximal processing rate μ^e measured in parts per unit time. The density of products $\rho^e(x,t,\omega)$ is governed by the continuity equation for all $x \in [0,1]$, $t \geq 0$, $\omega \in \Omega$:

$$\partial_t \rho^e(x,t,\omega) + \partial_x f^e(\rho^e(x,t,\omega)) = 0, \quad \rho(x,0,\omega) = \rho_0^e(x), \tag{2.4a}$$

where the flux function f^e is given by

$$f^e(\rho^e) = \min\Big\{ v^e \cdot \rho^e(x,t,\omega), \mu^e \cdot r^e(t,\omega) \Big\}. \tag{2.4b}$$

In particular, this means that, if the processor e is not broken, i.e. $r^e = 1$, then the density of goods $\rho^e(x,t,\omega)$ is transported with velocity v^e and the flux is less than or equal to the maximal processing rate μ^e. On the other hand, if the processor e is broken, i.e. $r^e = 0$, then no goods are processed at all and the flux is zero.

Each processor e has the possibility of storing goods that cannot be processed immediately in a queue $q^e(t,\omega)$ (see Figure 2.2).

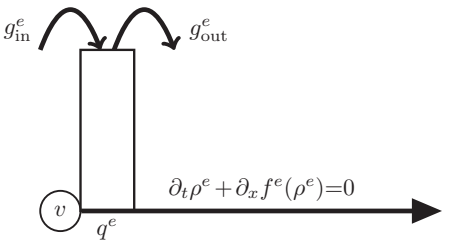

FIG. 2.2. *A processing unit is composed of an ordinary differential equation describing the load of a queue q^e coupled to the dynamics of the processor governed by a conservation law.*

The inflow to this queue is denoted by the function $g^e_{\text{in}}(t,\omega)$ and the outflow of the queue by the function $g^e_{\text{out}}(t,\omega)$. The dynamics of the queue are determined by the difference between its inflow $g^e_{\text{in}}(t,\omega)$ and its outflow $g^e_{\text{out}}(t,\omega)$. Thus, the load of the queue $q^e(t,\omega)$ is given by the rate equation

$$\partial_t q^e(t,\omega) = g^e_{\text{in}}(t,\omega) - g^e_{\text{out}}(t,\omega), \quad q^e(0,\omega) = q^e_0. \tag{2.5}$$

For the inflow to the queue $g^e_{\text{in}}(t,\omega)$, we remark that, if the origin $s(e)$ of processor e is an inflow point to the network, i.e. $s(e) \in \mathcal{V}_{\text{in}}$, the inflow is given by the inflow function $G^v_{\text{in}}(t)$. On the other hand, if the origin of processor e is an inner vertex, i.e. $s(e) \notin \mathcal{V}_{\text{in}}$, the inflow is given by the sum of all incoming flows multiplied by the distribution or routing parameter $A^{s(e),e}(t)$, which describes the percentage of flow sent to processor e (cf. Figure 2.3). Routing parameters are a degree of freedom in simulation models and will be the controls for optimization purposes in Section 3.

For a vertex v and any outgoing processor $e \in \delta_v^+$, the routing parameters $A^{v,e}$ are defined as follows.

DEFINITION 2.1 (Distribution rates). *For any vertex $v \in \mathcal{V}$ with $|\delta_v^+| \neq \emptyset$ and any processor $e \in \delta_v^+$, the distribution rate $A^{v,e}(t)$ should fulfill two conditions for all $t \geq 0$:*

(i) $0 \leq A^{v,e}(t) \leq 1$.

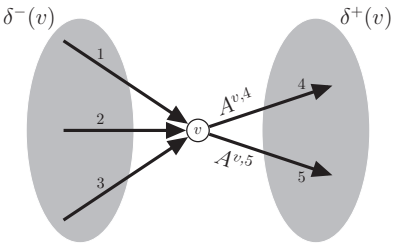

FIG. 2.3. *Illustrations of* $\delta_v^{\pm}$ *as well as of the distribution rates* $A^{v,4}(t)$ *and* $A^{v,5}(t)=1-A^{v,4}$, *respectively.*

(ii) $\sum_{e\in\delta_v^+} A^{v,e}(t)=1$.

Now, we are able to replace the inflow $g_{\text{in}}^e(t,\omega)$ in Equation (2.5) with

$$g_{\text{in}}^e(t,\omega)=\begin{cases} A^{s(e),e}(t)\sum\limits_{\bar{e}\in\delta_{s(e)}^-} f^{\bar{e}}(\rho^{\bar{e}}(1,t,\omega)) & \text{if } s(e)\notin\mathcal{V}_{\text{in}},\\ G_{\text{in}}^{s(e)}(t) & \text{if } s(e)\in\mathcal{V}_{\text{in}}. \end{cases} \tag{2.6}$$

The outflow $g_{\text{out}}^e(t,\omega)$ appearing in Equation (2.5) can also be specified: If the queue $q^e(t,\omega)$ is empty, the outflow equals the minimum of the ingoing flow $g_{\text{in}}^e(t,\omega)$ and the maximal capacity $\mu^e\cdot r^e(t,\omega)$ of the processor. If the queue is filled, the queue is reduced with maximal capacity of the processor. This yields

$$g_{\text{out}}^e(t,\omega)=\begin{cases} \min\{g_{\text{in}}^e(t,\omega),\mu^e\cdot r^e(t,\omega)\} & \text{if } q^e(t)=0,\\ \mu^e\cdot r^e(t,\omega) & \text{if } q^e(t)>0. \end{cases} \tag{2.7}$$

Summarizing, the stochastic simulation network model is given by the equations:

$$\Big\{ \text{ (2.2), (2.4), (2.5), (2.6), (2.7).} \tag{2.8}$$

Note that the stochastic network model (2.8) also captures a deterministic production network. Therefore, we initialize all processors $e\in\mathcal{A}$ with states $r_0^e=1$ and set $\tau_{\text{on}}^e=\infty$. Consequently, the processors do not switch and keep operating for the whole time. This fact will be used in the numerical experiments in Section 4.

2.2. Routing strategies. A crucial point in production network models is the distribution of incoming goods among the outgoing processors (cf. Equation (2.5) and Equation (2.6)). In the following, we introduce two different types of control strategies for the simulation model (2.8) that are compared to solutions obtained by solving an optimization model in Section 4. In our approach, we distinguish between state-independent (shortly s-i) and state-dependent (shortly s-d) routing strategies that take the current states of the processors into account. Certainly, all strategies or distribution rates need to fulfill the properties stated in Definition 2.1. In the following, we will write $\alpha^{s(e),e}$ instead of $A^{s(e),e}(t)$ whenever the distribution rates are time-independent.

2.2.1. State-independent strategies. Göttlich et al. [22] consider a very intuitive way to control the incoming flow among outgoing processors, i.e. where the flow is equally distributed according to the number of outgoing processors. We call this strategy the *s-i uniform control* strategy.

DEFINITION 2.2 (s-i uniform control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$, i.e. $v = s(e)$, the ongoing processors. Then, we define the s-i uniform control by*

$$\alpha_{uniform}^{s(e),e} = \frac{1}{|\delta_{s(e)}^+|}. \tag{2.9}$$

Note that the s-i uniform control (2.9) is time-independent since δ_v^+ only depends on the topology of the network.

Sticking to topology-dependent and time-independent constant controls, we define the *s-i capacity control* strategy. Here, the distribution rates are proportional to the maximal capacities μ^e of outgoing processors.

DEFINITION 2.3 (s-i capacity control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with capacity μ^e. Then, we define the s-i capacity control by*

$$\alpha_{capacity}^{s(e),e} = \frac{\mu^e}{\sum\limits_{\bar{e} \in \delta_{s(e)}^+} \mu^{\bar{e}}}. \tag{2.10}$$

These rates are again time-independent, because the maximal capacities μ^e are constant parameters and will not change in time.

In the presence of random breakdowns of processors, the mean availability of each processor is another important characteristic that should be included in the routing strategy. For a single processor $e \in E$, the mean availability can be computed by

$$\tau_{\text{mean}}^e = \frac{\tau_{\text{on}}^e}{\tau_{\text{on}}^e + \tau_{\text{off}}^e},$$

where $\tau_{\text{on}}^e > 0$ is the mean time between failures and $\tau_{\text{off}}^e > 0$ is the mean repair time (cf. Figure 2.1). For a processor e which is not considered to break down, we set $\tau_{\text{mean}}^e = 1$. Including the mean availability, we end up with a more elaborate routing strategy taking both the capacity and the availability into account.

DEFINITION 2.4 (s-i availability control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with the capacity μ^e and mean availability τ_{mean}^e. Then, we define the s-i availability control by*

$$\alpha_{availability}^{s(e),e} = \frac{\mu^e \cdot \tau_{mean}^e}{\sum\limits_{\bar{e} \in \delta_{s(e)}^+} \mu^{\bar{e}} \cdot \tau_{mean}^{\bar{e}}}. \tag{2.11}$$

Note that strategy (2.11) is also time-indepedent because the mean availabilities do not change in time.

Another state-independent, but now time-dependent, strategy might additionally depend on the queue load. Let the *inverse relative queue load* of a processor $e \in E$ be defined by

$$q_{\text{rel}}^e(t) = \begin{cases} \mu^e / q^e(t) & \text{if } q^e(t) > \mu^e, \\ 1 & \text{else}, \end{cases}$$

where $q^e(t)$ is the current queue load and μ^e the maximum capacity. This leads to the property $0 < q^e_{\text{rel}}(t) \leq 1$ since all processors have positive maximal capacities $\mu^e > 0$. So, the relative queue load $q^e_{\text{rel}}(t)$ can be seen as the percentage of queue load which is cleared from the queue in one time step. Clearly, the lower the value of $q^e_{\text{rel}}(t)$ gets, the worse is the routing of goods.

This gives rise to another routing strategy extending the former state-independent strategies by relative queue loads.

DEFINITION 2.5 (s-i queuing control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with the capacity μ^e, mean availability τ^e_{mean} and relative queue load $q^e_{rel}(t)$ for $t \geq 0$. Then, we define the s-i queuing control by*

$$A^{s(e),e}_{queue}(t) = \frac{\mu^e \cdot \tau^e_{mean} \cdot q^e_{rel}(t)}{\sum\limits_{\bar{e} \in \delta^+_{s(e)}} \mu^{\bar{e}} \cdot \tau^{\bar{e}}_{mean} \cdot q^{\bar{e}}_{rel}(t)}. \tag{2.12}$$

Note that the distribution strategy (2.12) is time-dependent but still independent from the state of processors.

2.2.2. State-dependent strategies. State-dependent strategies compared to state-independent strategies are mainly concerned with random breakdown scenarios. For instance, if a processor is broken at time t, e.g. $r^e(t) = 0$, goods should no longer be fed into this processor since otherwise they pile up and the corresponding queue starts to increase. For this reason, we are now interested in state-dependent (or s-d) strategies that do not distribute goods into broken processors unless all ongoing processors are broken. This is done by converting state-independent (cf. Section 2.2.1) into state-dependent strategies. We start with the uniform control.

DEFINITION 2.6 (s-d uniform control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with state $r^e(t)$ for $t \geq 0$. Then, we define the s-d uniform control by*

$$A^{s(e),e}_{uniform}(t) = \begin{cases} 1 / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) > 0, \\ 1 / |\delta^+_{s(e)}| & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) = 0. \end{cases} \tag{2.13}$$

Here, in the first case the sum indicates the number of processors $e \in \delta^+_{s(e)}$ which are operating at time $t \geq 0$.

In the same way, we design state-dependent counterparts of Equation(2.10), Equation (2.11), and Equation (2.12). According to the s-d uniform control (2.13), we introduce a second case that resembles the state-independent control if all ongoing processors are down.

DEFINITION 2.7 (s-d capacity control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with the capacity μ^e and state $r^e(t)$ for $t \geq 0$. Then, we define the s-d capacity control by*

$$A^{s(e),e}_{capacity}(t) = \begin{cases} \left(\mu^e \cdot r^e(t)\right) / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} \left(\mu^{\bar{e}} \cdot r^{\bar{e}}(t)\right) & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) > 0, \\ \mu^e / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} \mu^{\bar{e}} & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) = 0. \end{cases} \tag{2.14}$$

DEFINITION 2.8 (s-d availability control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with the capacity μ^e, mean availability τ^e_{mean}, and state $r^e(t)$ for $t \geq 0$. Then, we define the s-d availability control by*

$$A^{s(e),e}_{availability}(t) = \begin{cases} \left(\mu^e \cdot \tau^e_{mean} \cdot r^e(t)\right) / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} \left(\mu^{\bar{e}} \cdot \tau^{\bar{e}}_{mean} \cdot r^{\bar{e}}(t)\right) & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) > 0, \\ \left(\mu^e \cdot \tau^e_{mean}\right) / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} \left(\mu^{\bar{e}} \cdot \tau^{\bar{e}}_{mean}\right) & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) = 0. \end{cases} \tag{2.15}$$

DEFINITION 2.9 (s-d queuing control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with the capacity μ^e and mean availability τ^e_{mean}. Furthermore, let $q^e_{rel}(t)$ be the relative queue load and $r^e(t)$ the state for each processor $e \in \delta_v^+$ and $t \geq 0$. Then, we define the s-d queuing control by*

$$A^{s(e),e}_{queuing}(t) = \begin{cases} \left(\mu^e \cdot \tau^e_{mean} \cdot q^e_{rel}(t) \cdot r^e(t)\right) / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} \left(\mu^{\bar{e}} \cdot \tau^{\bar{e}}_{mean} \cdot q^{\bar{e}}_{rel}(t) \cdot r^{\bar{e}}(t)\right) & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) > 0, \\ \left(\mu^e \cdot \tau^e_{mean} \cdot q^e_{rel}(t)\right) / \sum\limits_{\bar{e} \in \delta^+_{s(e)}} \left(\mu^{\bar{e}} \cdot \tau^{\bar{e}}_{mean} \cdot q^{\bar{e}}_{rel}(t)\right) & \text{if } \sum\limits_{\bar{e} \in \delta^+_{s(e)}} r^{\bar{e}}(t) = 0. \end{cases} \tag{2.16}$$

Finally, additional to the state-dependent strategies (2.13)–(2.16), we consider the following non-trivial adaption. The idea is to allocate goods also to an in-operating processor as long as the relative queue load is smaller than a constant c. If this processor is on again, it can start directly at its maximum capacity, which might be higher than its average workload. We expect that this strategy helps to regain the lost time when the processor was off. Therefore, we intend to distribute the flow among processors e that are in progress, i.e. $r^e(t) = 1$, and have a relative queue load of at least c, i.e. $q^e_{\text{rel}} > c$ (cf. cases 2 and 3 in Equation (2.17)). In the case where all processors are down or the relative queue load is smaller than c (case 1 in Equation (2.17)), we distribute the flow according to Definition 2.5.

DEFINITION 2.10 (advanced s-d control). *Let $v \in \mathcal{V}$ be an inner vertex and $e \in \delta_v^+$ the ongoing processors with the capacity μ^e and mean availability τ^e_{mean}. Furthermore, let $q^e_{rel}(t)$ be the relative queue load and $r^e(t)$ the state for each processor $e \in \delta_v^+$ and $t \geq 0$. Then, we define the advanced s-d control with the user-defined constant $0 \leq c \leq 1$ by*

$$A^{s(e),e}_{adv}(t) = \begin{cases} A^{s(e),e}_{queue}(t) & \text{if } \forall e \in \delta^+_{s(e)} : r^e(t) = 0 \text{ or } q^e_{rel}(t) < c, \\ \frac{\mu^e \cdot \tau^e_{mean} \cdot q^e_{rel}(t)}{\sum_{\bar{e} \in \delta^+_{s(e)}, q^{\bar{e}}_{rel}(t) > c, r^{\bar{e}}(t) = 1} \left(\mu^{\bar{e}} \cdot \tau^{\bar{e}}_{mean} \cdot q^{\bar{e}}_{rel}(t)\right)} & \text{if } r^e(t) = 1 \text{ and } q^e_{rel}(t) > c, \\ 0 & \text{else.} \end{cases} \tag{2.17}$$

We remark that the the parameter $0 \leq c \leq 1$ switches between state-dependent and -independent controls. For instance, $c = 0$ reproduces the s-d queuing control (2.16) and $c = 1$ the s-i queuing control (2.12). Preliminary tests have shown that, for the choice $c = 1/2$, the results of stratgey (2.17) lie inbetween the results of the bounding strategies (2.12) and (2.16), with a tendency to the better-performing one. For that reason, the computations in Section 4 are performed using $c = 1/2$.

2.3. Numerical solution method. We remark that the stochastic network model (2.8) is piecewise deterministic, i.e. there occurs no further stochasticity between the switching points. This property is in particular used to solve model (2.8) numerically while following the idea of Gillespie [17, 18], who first studied piecewise deterministic processes (PDPs). The proposed Stochastic Simulation Algorithm (SSA) has been successfully adapted to production networks with random breakdowns [22] and unreliable flow lines [21]. For our simulation purposes, we use the version from [22] and modify Algorithm 4.2 therein to fit to our routing strategies defined above (cf. Algorithm 1).

Algorithm 1 Pseudocode: adapted stochastic sampling algorithm

Input: $[0,T]$ real interval. Initial data for $t=0$. Strategy for distribution rates.
Output: Simulation of the production network for one realization ω of model (2.8) on $[0,T]$.

1: **while** $t_i^* < T$ **do**
2: Sample next switching point t_{i+1}^*.
3: Compute solution in the interval $\left[t_i^*, t_{i+1}^*\right)$ using Euler steps for the updates of the queues and an upwind scheme for the update on the arcs. The distribution rates at the vertices are given by the predefined strategy.
4: Set $i = i+1$.
5: **end while**

3. An approximate optimization algorithm

We intend to compare the distribution strategies discussed in Section 2.2 to approximate solutions to the following optimization problem:

$$\begin{aligned} &\max_{A^{v,e}} / \min_{A^{v,e}} J(\rho) \\ &\text{s.t. } (2.8), \end{aligned} \tag{3.1}$$

where we consider two different objective functions. On the one hand, the approximate maximization of throughput corresponds to the approximate maximization of flow at the end of the processors $e \in E^{\text{out}}$ leading to a sink, i.e.

$$\max_{A^{v,e}} J(\rho) = \int_{t_i^*}^{t_i^*+\tau_{\text{hor}}} \sum_{e \in E^{out}} f^e\left(\rho^e(1,t,\omega)\right)\, \mathrm{d}t. \tag{3.2}$$

This choice of objective function only aims at high outputs at exiting arcs but disregards the queue loads inside the network. To tackle this problem, we consider a second objective, namely the approximate minimization of queues

$$\min_{A^{v,e}} J(\rho) = \sum_{e \in E} \frac{1}{2} \int_{t_i^*}^{t_i^*+\tau_{\text{hor}}} \left(q^e(t)\right)^2 \, \mathrm{d}t. \tag{3.3}$$

This type of objective not only minimizes the total amount of stored goods but also maximizes the throughput at the same time. We choose a quadratic formulation in Equation (3.3) as in [20] to achieve better convergence.

Apparently, problem (3.1) is constrained by differential equations, i.e. the stochastic production model (2.8), and the controls to be determined are the distribution rates $A^{v,e}$.

As we have seen in Section 2, model (2.8) is accompanied by stochastic processes, i.e. the random failures of processors, and therefore the optimization model (3.1) is also stochastic. To exploit the underlying piecewise deterministic structure of the model equation (2.8), we solve the control problem on a rolling time horizon. That means, at each switching point t_i^*, we fix the states r^e of the processors e for the time interval $[t_i^*, t_i^* + \tau_{\text{hor}}]$, where $\tau_{\text{hor}} > 0$ is a prediction parameter defining the length of the considered time horizon. For this time interval, we optimize the objective function leading to constant distribution rates in the interval $[t_i^*, t_i^* + \tau_{\text{hor}}]$. The utilized numerical algorithm to solve the optimization problem (3.1) is the MATLAB routine *fmincon* [29]. We repeat this procedure for each switching point t_i^* to obtain a solution on $[0,T]$ (cf. algorithms 2, 3). Consequently, this leads to an approximate optimization algorithm, as the computational solution depends on the prediction parameter τ_{hor}.

Algorithm 2 Pseudocode: rolling time horizon

Input: $[0,T]$ real interval. Initial data for $t = 0$.
Output: Distribution rates for one realization ω of model (2.8) on $[0,T]$.

```
while t_i^* < T do
    Sample next switching point t_{i+1}^*.
    Compute solution in the interval [t_i^*, t_i^* + τ_hor) using Algorithm 3.
    Set i = i + 1.
end while
```

Algorithm 3 Pseudocode: approximate solution for one time interval

Input: t_i^* sampled switching times, τ_{hor} time horizon, $r^e(t_i^*)$ states of the processors $e \in E$, and $\rho^e(t)$ and $q^e(t)$ for $0 \le t \le t_i^*$ and for all processors $e \in E$.
Output: $A^{v,e}(t)$, $\rho^e(t)$, and $q^e(t)$ for $0 \le t \le t_i^* + \tau_{\text{hor}}$ and for all $e \in E$.

1: Fix states $\bar{r}^e(t) = r^e(t_i^*)$ for $t \ge t_i^*$.
2: Set $\alpha_0^{v,e} = A^{v,e}(t_i^*)$ as initial distribution rates.
3: Apply *fmincon* to solve Equation (3.1) for $\alpha_{\text{opt}}^{v,e}$ within $[t_i^*, t_i^* + \tau_{\text{hor}}]$ with $r^e = \bar{r}^e$ using $\alpha_0^{v,e}$ as initial rates.
4: Set $A^{v,e}(t) = \alpha_{\text{opt}}^{v,e}$ for $\bar{t}_i^* \le t \le t_i^* + \tau_{\text{hor}}$.

Note that the time horizon τ_{hor} should be chosen large enough to provide reliable results. If τ_{hor} is chosen too small, i.e. smaller than the longest path from a source to a sink, no flow is distributed along this path. Furthermore, τ_{hor} must be larger than the mean time between the switchings to ensure the definition of $A^{v,e}(t)$ for all t (cf. Algorithm 3, Step 4).

4. Numerical results

In this section, we present the numerical results comparing the different routing heuristics and the approximate optimization procedures proposed in Section 2.2 and 3, respectively. We start with a qualitative study of different network topologies since the routing highly depends on the underlying geometry and parameter configuration. To cover reasonable settings, we distinguish between three different types of networks: the diamond network, the cascade network, and a non-symmetric network. They are typically characterized by their size and capacity allocation. Another crucial ingredient in our experiments is the choice of the arrival rates (or inflow functions). For each network sample, we consider the following two scenarios: The first inflow function is constant

for all times and fixed to 80% of the network's deterministic bottleneck capacity μ^{e_b}, whereas the second inflow function repeats a cycle of delivering 100% for 30 time steps and stopping the inflow for 10 time steps for five times. Consequently, we refer to the two functions as *constant inflow* and *stop-go inflow*, respectively. The constant inflow is used to analyze how the proposed routing strategies behave for non-time varying filling of the system. In contrast, the stop-go inflow is highly fluctuating to see how the routing strategies react and respond on variations in time (cf. figures 4.1 and 4.2). Note that the total constant inflow $\int_0^T G_{\text{in}}^{v_1}(t)\,\mathrm{d}t = 0.8\mu^{e_b}\cdot T$ is 5% larger than the total stop-go inflow $\int_0^T G_{\text{in}}^{v_1}(t)\,\mathrm{d}t = 0.75\mu^{e_b}\cdot T$.

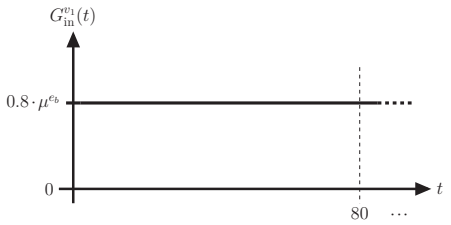

FIG. 4.1. *Constant inflow profile.*

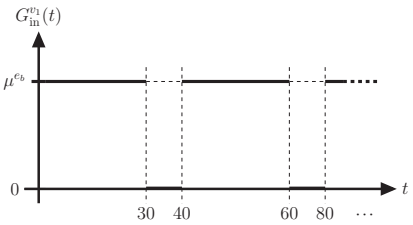

FIG. 4.2. *Stop-go inflow profile.*

We close the section with a comparison of computation times, where we particularly analyze the influence of the network size and its parameters. To fairly compare the running times, all simulations and optimizations have been performed on a PC equipped with 32GB RAM, Intel(R) Xeon(R) CPU E3-1280 @ 3.60GHz.

For our investigations, we use a spatial discretization of $\Delta x = 1/9$ to solve the model (2.8) and (3.1) as well. Setting the velocities to $v^e = 1$ for all processors, we end up with the step size $\Delta t = 1/9$ due to the CFL condition $\Delta t \le \Delta x$. For all test cases, we consider a total time horizon of $T = 200$.

4.1. Diamond network. The first network to be considered is the so-called *diamond network* consisting of 8 arcs (cf. Figure 4.3. Here, the capacities of each processor e are illustrated together with its index. We also color-coded the availability of the processors. The green arcs represent deterministic processors with availability $\tau_{\text{mean}}^e = 1$. We marked the processors (arcs) slightly failing (with an availability of $\tau_{\text{mean}}^e = 95\%$) in yellow and the ones most prone to failure (availability $\tau_{\text{mean}}^e = 75\%$) in red (cf. Table 4.1). The diamond network consists of two vertices, namely v_3 and v_4, where the flux has to be controlled according to a routing policy or approximate optimization.

First, we present the results for the constant influx, i.e. we consider for all times t

$$G_{\text{in}}^{v_1}(t) = 32 \tag{4.1}$$

at processor $e = 1$. This equals 80% of the network's deterministic bottleneck capacity (cf. Figure 4.1).

In Figure 4.4, the total outflow of the network for one sample $\omega \in \Omega$

$$\int_0^T \sum_{e \in E^{\text{out}}} f^e(\rho^e(1,t,\omega))\,\mathrm{d}t \tag{4.2}$$

at $T = 200$ is shown for all nine heuristic routing strategies (blue squares, strategies (2.9)–(2.17)) and the two approximate optimization approaches (white squares,

processors		parameters		
type	indices	availability τ^e_{mean}	mean up τ^e_{on}	mean down τ^e_{off}
green	1,8	1	∞	0
yellow	2,4,6,7	0.95	47.5	2.5
red	3,5	0.75	30	10

TABLE 4.1. *Parameters for the diamond network.*

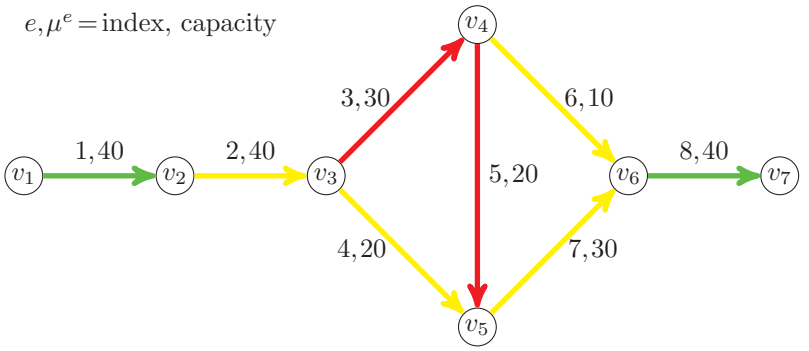

FIG. 4.3. *The diamond network. The values at the arcs describe the capacities of the processors. Processors with the same configuration share the same color.*

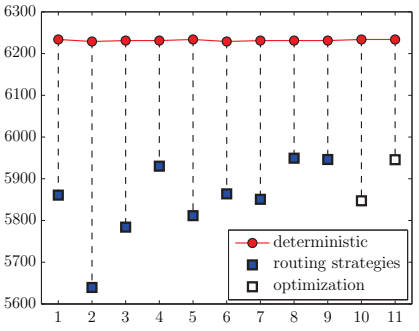

FIG. 4.4. *Total outflow of the diamond network at $T = 200$ for all control strategies and constant inflow.*

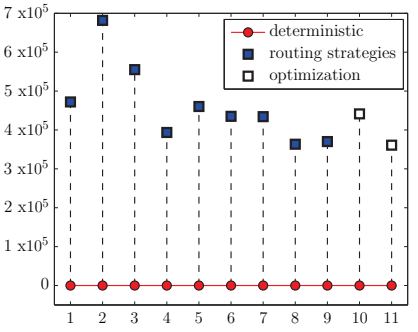

FIG. 4.5. *Sum of all queues within the diamond network for all control strategies and constant inflow.*

strategies (3.2) and (3.3)). To improve readability, we use this notation throughout this section. Furthermore, we provide results for the deterministic model (red dots) as a benchmark solution for the aforementioned computations. In this way, we directly observe the influence of the stochasticity on the single strategies.

The sum of all queues within the network over the whole time horizon and for sample $\omega \in \Omega$

$$\sum_{e=1}^{M} \int_0^T q^e(t,\omega)\, \mathrm{d}t \tag{4.3}$$

is presented in Figure 4.5 using the same order as above. The maximal queue length

$$\max_{e\in E}\max_{0\le t\le T} q^e(t,\omega) \tag{4.4}$$

arising at one processor within the network is presented in Table 4.2.

The second inflow function we consider for $0\le t\le T$ is plotted in Figure 4.2 and given by

$$G_{\text{in}}^{v_1}(t)=\begin{cases}40 & \text{if } 0\le \text{mod}(t,40)<30,\\ 0 & \text{if } 30\le \text{mod}(t,40)<40.\end{cases} \tag{4.5}$$

In figures 4.6 and 4.7 and Table 4.3, again the total outflow of the network at $T=200$, the sum of all queues and the maximal occurring queue lengths are presented.

objective	strategy										
	1	2	3	4	5	6	7	8	9	10	11
maximum queue	162	287	205	98	292	181	233	92	98	188	68

TABLE 4.2. *Maximum queues of the diamond network for a constant inflow.*

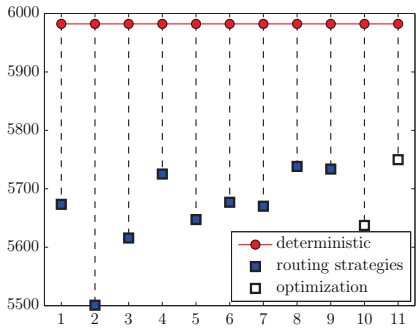

FIG. 4.6. *Total outflow of the diamond network at $T=200$ for all control strategies and stop-go inflow.*

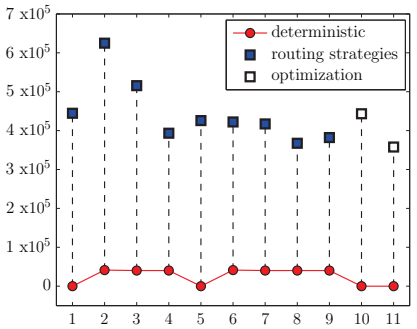

FIG. 4.7. *Sum of all queues within the diamond network for all control strategies and stop-go inflow.*

objective	strategy										
	1	2	3	4	5	6	7	8	9	10	11
maximum queue	168	276	206	103	274	190	232	117	99	202	96

TABLE 4.3. *Maximum queues of the diamond network for a stop go inflow.*

From figures 4.4 and 4.5, we see that, in the deterministic case, all strategies work equally well for the constant inflow (4.1). Additionally, queues remain empty. This means the available capacity of the network is sufficient to process all incoming parts. This changes for the stop-go inflow (4.5), where queues start to build up for various cases. For both scenarios, the strategies in the stochastic case lead to completely different results. Obviously, the s-i capacity control (2.10) is worst in all three performance

measures and for both inflow functions. This is due to processors $e=3,5$ both having higher capacity but lower availability than processors $e=4,6$. Consequently, the flow is mainly distributed in those processors and gets stuck more often. The best performing strategies for both inflows and for all three performance measures are s-i queuing (2.12), s-d queuing (2.16) as well as the advanced s-d strategy (2.17). Note that the latter is a mixture of the earlier ones.

We also remark that the state-dependent strategies (2.16) and (2.17) work slightly better than the state-independent strategy (2.12), but the difference is rather small. Nevertheless, the state-dependent strategies yield better results compared to state-independent strategies. This is also true for the capacity controls (cf. for instance s-d capacity control (2.14)) with s-i capacity control (2.10). For all other control pairs (state-independent vs. state-dependent), the state-independents control (strategies (2.9) and (2.11)) perform much better than their respective state-dependent analogues (strategies (2.13) and (2.15)). The maximum queue values in Table 4.3 are important for the design of inventories. However, they are observed at single points in time and do not reflect the total utilization in the considered time horizon.

Concerning the approximate optimization approaches, we detect that the quadratic objective function (3.3) yields the best overall performance. We note that the optimization problem (3.1) with the objective function (3.2) performs even worse than the best heuristics. This is due to the state-dependency of the approximate optimization algorithm and the fact that the extrapolation from the current point in time may lead to a bad prediction in the worst case.

4.2. A cascade network. The second network to be considered is a large symmetric cascade network with 27 arcs shown in Figure 4.8. The main ingredient of this network are the first and second layer of processors ($e=6,\dots,23$) consisting of nine processors each. In the first layer, three processors lead from each of the three vertices v_3, v_4, and v_5 to each of the three vertices v_6, v_7, and v_8 of the second layer. Hereby, the processors heading "straight" down are more prone to failure than all the other processors of this layer. In the second layer, the processors leaving vertices v_6 and v_8 are those most likely to fail. As depicted in Table 4.4, the first five and last four processors are deterministic and do not fail.

As before, we first present results for the constant inflow (cf. Figure 4.1), which is now

$$G_{\text{in}}^{v_1}(t)=72 \tag{4.6}$$

in figures 4.9 and 4.10 and Table 4.5. The performance measures are again the total outflow of the network (4.2), the sum of queues within the network (4.3) and the maximal queue lengths (4.4).

Second, we consider the stop-go-inflow for $0\le t\le 200$ (cf. Figure 4.2)

$$G_{\text{in}}^{v_1}(t)=\begin{cases}90 & \text{if } 0\le \operatorname{mod}(t,40)<30,\\ 0 & \text{if } 30\le \operatorname{mod}(t,40)<40\end{cases} \tag{4.7}$$

applied to the cascade network and show the results in figures 4.11 and 4.12 and Table 4.6.

Due to the full symmetry of the network, the uniform and capacity control strategy yield the same results for the state-independent and -dependent cases, respectively. From figures 4.9 and 4.11, we see that, despite the availability control (strategy (2.15)), the state-dependent controls perform better than their state-independent analogues

when considering the total outflow of the network. While the performance of the different strategies in terms of the total outflow (4.2) is not significant for the constant inflow (4.6), we observe a change for the stop-go inflow (4.7). The controls also differ concerning the sum of all queues (4.3) for both inflow functions (cf. figures 4.10 and 4.12). The tendency is once more that the state-dependent controls perform better than the corresponding state-independent ones. However, the state-independent strategies lead to acceptable limits of the maximal queue length (4.4) (cf. tables 4.5 and 4.6). This is due to the choice of the availability of the processors in the first and second layer. Since the state-dependent controls try to avoid uncertain processors, more flow is led to vertices v_6 and v_8. Consequently, in the next processors, which are all uncertain, the queues start to increase. This drawback is avoided by the state-independent controls. Additionally, we observe that the maximal queue length of the state-independent controls arises at processors of the first layer (those leaving v_3, v_4, and v_5) while the state-dependent controls have maximal queues at processors of the second layer (those leaving v_6, v_7 and v_8). For the stop-go inflow (4.7), the sum of queues as well as the maximal queue lengths are smaller than for the constant inflow (4.6). The stop-go inflow also favors the queuing controls and the advanced control over the uniform controls. Those controls are able to exploit the stopping in the inflow to clear the queues.

processor		parameters		
type	indices	τ^e_{mean}	τ^e_{on}	τ^e_{off}
green	1–5, 24–27	1	1	0
yellow	6,8,9,11,12,14,18–20	0.95	47.5	2.5
red	7,10,13,15–17,21–23	0.75	30	10

TABLE 4.4. *Parameters for the cascade network.*

objective	strategy										
	1	2	3	4	5	6	7	8	9	10	11
maximum queue	258	258	333	332	296	296	363	325	333	145	150

TABLE 4.5. *Maximum queues of the cascade network for a constant inflow.*

objective	strategy										
	1	2	3	4	5	6	7	8	9	10	11
maximum queue	229	229	282	220	252	252	305	228	218	156	128

TABLE 4.6. *Maximum queues of the cascade network for a stop go inflow.*

Different from the diamond network example, the approximate optimization (3.2) yields an improvement of $\geq 6\%$ in the total outflow, of $\geq 80\%$ for the sum of all queues and of $\geq 70\%$ for the maximum queue length in the case of constant inflow (4.6). This shows that in contrast to the heuristics strategies, which consider only local criteria for the distribution of flow, the approximate optimization process includes global information of the network. Thus, the approximation algorithm is able to detect and avoid uncertain processors in the second layer that the heuristics do not to see. For the stop-go inflow (4.7), the approximate optimization still performs better than the heuristics, but the benefit is less ($\geq 2\%$, $\geq 25\%$, and $\geq 40\%$, respectively).

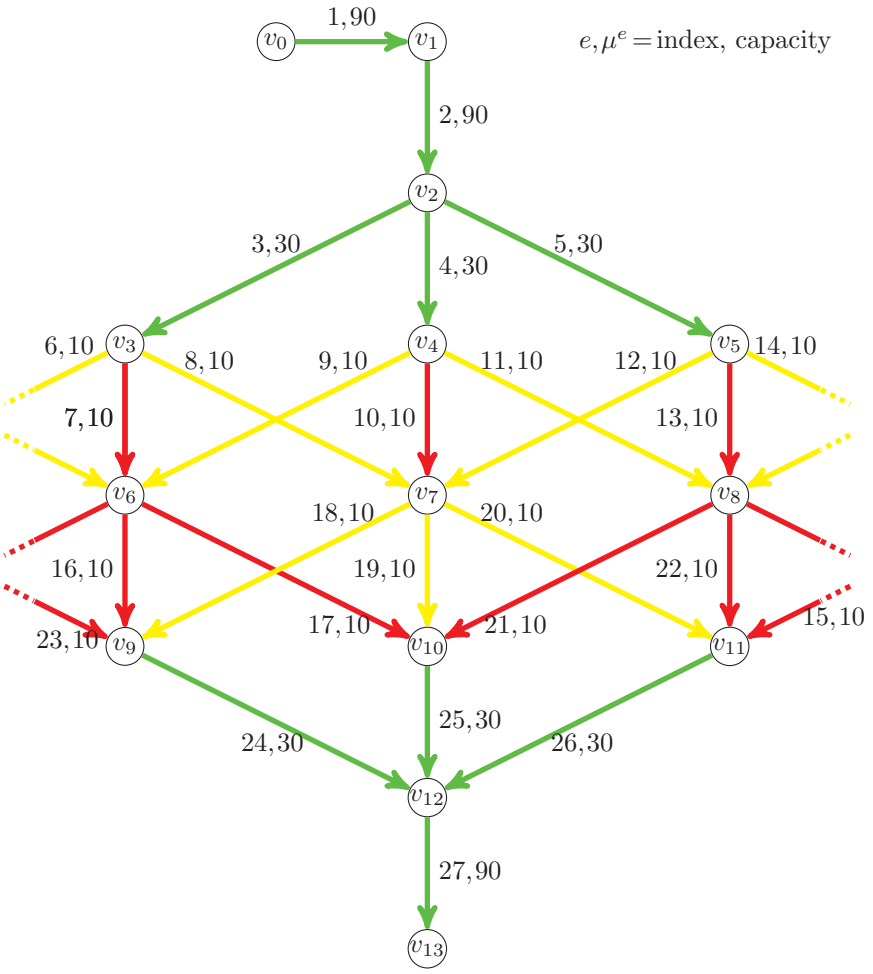

FIG. 4.8. *The cascade network.*

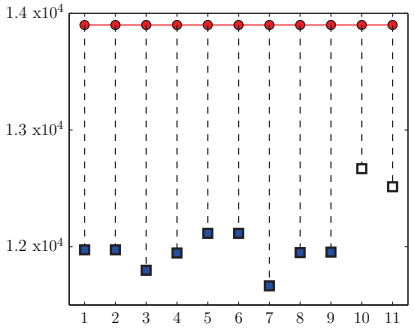

FIG. 4.9. *Total outflow of the cascade network at $T = 200$ for all control strategies and constant inflow.*

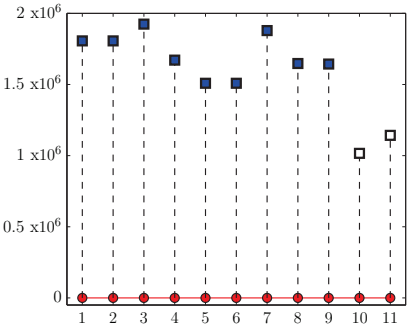

FIG. 4.10. *Sum of all queues within the cascade network for all control strategies and constant inflow.*

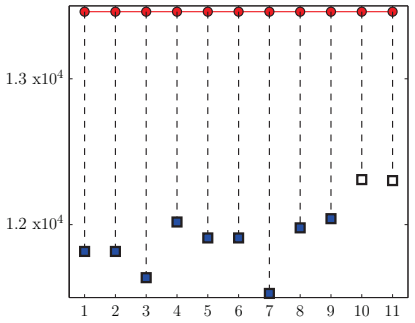

FIG. 4.11. *Total outflow of the cascade network at $T=200$ for all control strategies and stop-go inflow.*

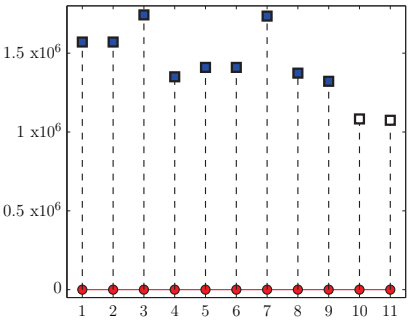

FIG. 4.12. *Sum of all queues within the cascade network for all control strategies and stop-go inflow.*

Note that, due to the time limit of 1 month (744 hours), only computational results for 14 optimization runs can be presented for the objective function (3.3). Therefore, we do not observe an improvement compared to Equation (3.2).

4.3. Non-symmetric network. Finally, we consider a non-symmetric network shown in Figure 4.13 with parameters given in Table 4.7. In this network, there is only one (deterministic) feeding processor $e=1$, and the processors leading to the sinks v_{10} and v_{11}, i.e. processors $e=17$ and $e=16$, are also reliable. Furthermore, the processors $e=3,4,7$ and $e=13$ are unreliable with an availability of $\tau^e_{\text{mean}}=0.75$, while all other processors have an availability of $\tau^e_{\text{mean}}=0.95$. The network is non-symmetric with respect to the number of linked processors at each vertex as well as their capacities (cf. figures 4.8 and 4.13). Nevertheless, the non-symmetric network is arranged in such a way that, in the deterministic case, all flow entering the network could be completely processed given appropriate distributions at vertices.

processor		parameters		
type	indices	τ^e_{mean}	τ^e_{on}	τ^e_{off}
green	1,16,17	1	1	0
yellow	2,5,6,8–12,14,15	0.95	47.5	2.5
red	3,4,7,13	0.75	30	10

TABLE 4.7. *Parameters for the non-symmetric network.*

As a constant inflow to the non-symmetric network, we consider (cf. Figure 4.1)

$$G^{v_1}_{\text{in}}(t)=80. \tag{4.8}$$

The numerical results for the total outflow (4.2), the sum of queues (4.3), and the maximum queue length for Equation (4.8) are presented in figures 4.14 and 4.15 and Table 4.8, respectively.

Lastly, we consider the non-symmetric network with the stop-go inflow (cf. Figure 4.2)

$$G^{v_1}_{\text{in}}(t)=\begin{cases}100 & \text{if } 0\le \text{mod}(t,40)<30,\\ 0 & \text{if } 30\le \text{mod}(t,40)<40\end{cases} \tag{4.9}$$

for $0 \le t \le 200$ and show the results in figures 4.11 and 4.12 and Table 4.6.

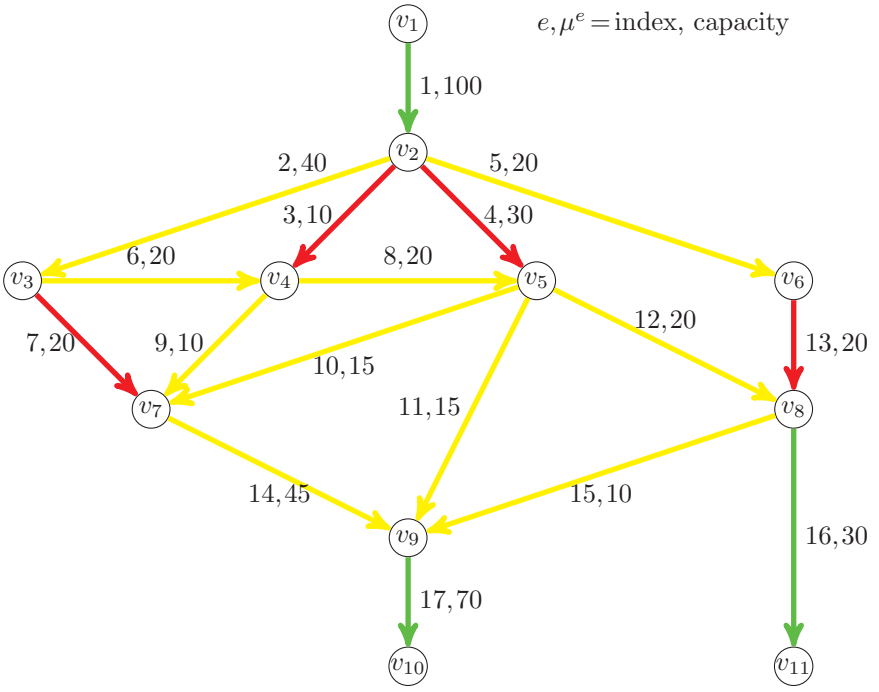

FIG. 4.13. *The non-symmetric network.*

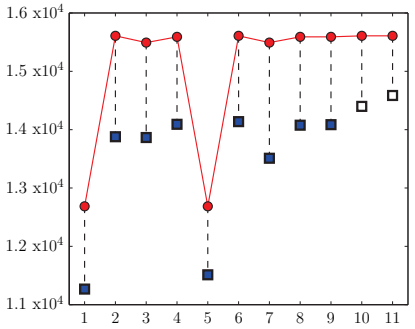

FIG. 4.14. *Total outflow of the non-symmetric network at $T = 200$ for all control strategies and constant inflow.*

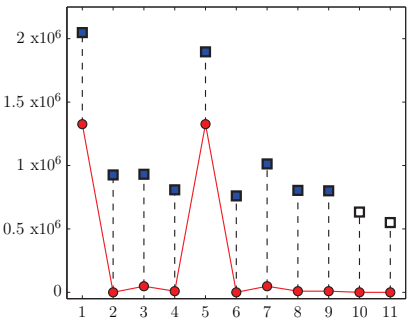

FIG. 4.15. *Sum of all queues within the non-symmetric network for all control strategies and constant inflow.*

From figures 4.14 – 4.17, we see that, while the deviation between the other strategies is small ($\le 10\%$), both the s-i uniform control (2.9) and the s-d uniform control (2.13) yield poor results for both inflow functions. We even recognize that, already in the deterministic case, those strategies perform worse than all other strategies in the stochastic regime. This is due to the non-symmetric structure of the network, where the uniform strategy naively distributes flow into capacity-restricted processors. For example, processor $e = 3$ is filled with the same amount of flow as processor $e = 2$ despite

$e=3$ having a quarter of the capacity of $e=2$ and a lower availability as well. The other strategies are able to avoid such situations and yield similar results with the queuing controls (2.12), (2.16), and, more preferable, the advanced control (2.17) strategy. For both inflow functions the availability controls (2.11) and (2.15) perform slightly worse than the average. But, for the stop-go inflow (4.9), we see that those controls perform best relative to their deterministic solution, since this result is already below average.

objective	strategy										
	1	2	3	4	5	6	7	8	9	10	11
maximum queue	2500	619	518	631	1849	584	861	635	634	277	231

TABLE 4.8. *Maximum queues of the non-symmetric network for a constant inflow.*

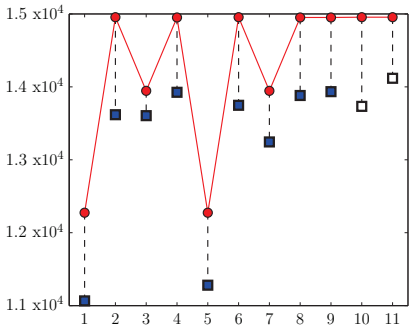

FIG. 4.16. *Total outflow of the non-symmetric network at $T=200$ for all control strategies and stop-go inflow.*

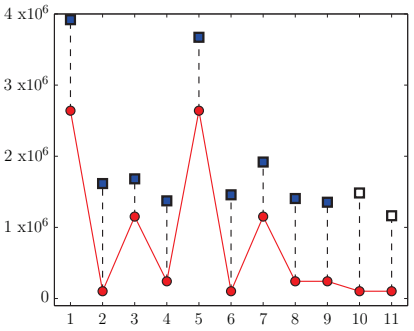

FIG. 4.17. *Sum of all queues within the non-symmetric network for all control strategies and stop-go inflow.*

objective	strategy										
	1	2	3	4	5	6	7	8	9	10	11
maximum queue	2327	535	426	452	1772	480	665	466	451	382	260

TABLE 4.9. *Maximum queues of the non-symmetric network for a stop-go inflow.*

Both approximate optimization approaches are able to outperform the heuristics for the constant inflow function (4.8). The approximate minimization of the queues in Equation (3.3) even performs slightly better than the approximate maximization of the outflow in Equation (3.2). As seen for the other networks before, the advantage of the approximate optimization algorithms becomes less for the stop-go inflow (4.9).

While the approximate maximization of the outflow (3.2) yields worse results concerning the total outflow and the queue loads compared to, for example, strategy (2.17), the approximate minimization of the queues in Equation (3.3) results in better objective values than all other strategies. Concerning the maximum queue lengths, both approximate algorithms are able to outperform the heuristics significantly ($\geq 10\%$).

4.4. Runtime analysis. To conclude our work, we compare the computing times for different network geometries and inflow patterns. We present runtimes for all strategies summed up over all Monte Carlo runs in Table 4.10. Note that the runtimes are given in hours.

topology			strategy										
network	MC-runs	influx	1	2	3	4	5	6	7	8	9	10	11
diamond	100	constant	0.33	0.32	0.32	0.34	0.34	0.35	0.36	0.36	0.37	16.6	16.6
		stop-go	0.33	0.33	0.32	0.35	0.35	0.35	0.36	0.37	0.37	16.6	16.6
cascade	30/14	constant	0.63	0.63	0.63	0.66	0.66	0.67	0.69	0.70	0.70	180	744
		stop-go	0.63	0.63	0.63	0.66	0.66	0.67	0.69	0.70	0.70	240	744
non-symmetric	30	constant	0.94	0.94	0.94	0.98	0.99	1.01	1.03	1.05	1.06	144	360
		stop-go	0.94	0.94	0.94	0.99	0.99	1.01	1.03	1.05	1.07	200	360

TABLE 4.10. *Total computation times in hours for according number of Monte Carlo runs.*

From Table 4.10, we see that the heuristic strategies are two to three orders of magnitude faster than the approximate optimization algorithms. While both approximations are equally slow when considering the diamond network, the approximate outflow maximization (3.2) provides significantly faster results than the approximate queue minimization (3.3) in the case of larger networks. The time limit for all computations was 1 month, i.e. 744 hours. Therefore, the approximate optimization algorithm (3.3) has been stopped after 14 runs only. Qualitatively, we observe that sometimes the flow maximizing approximation (3.2) performs worse than the best heuristic. In contrast to that, the queue minimizing approximation (3.3) performs better for almost all cases. While there is no difference in the runtime depending on the choice of the inflow function for the diamond network, the runtime increases drastically for the approximation algorithms using the stop-go inflow.

Comparing the computing times of the heuristic routing strategies (2.9)–(2.17), we see that they differ according to their time-dependence. More precisely, the state-independent controls (2.9), (2.10), and (2.11) (uniform, capacity, and availability, respectively) are only dependent on the network structure and can thus be computed in advance. This is time-efficient, and therefore those strategies are the fastest ones. The s-i queuing control (2.12) is independent of the state of processors but dependent on the relative queue load $q^e_{\mathrm{rel}}(t)$ at time t. Due to this time-dependence, it cannot be computed in advance. But obviously the computation is still faster than the state-dependent controls (2.13), (2.14), and (2.15) (uniform, capacity, and availability, respectively), which are the fastest state-dependent strategies. The most costly state-dependent controls are the s-d queuing (2.16) and advanced s-d control (2.17), which depend not only on the state but also on the relative queue length. The maximal deviation between the slowest and fastest computation is at most 14%. We point out that, in symmetric networks such as the diamond (cf. Figure 4.3, Section 4.1) or the cascade (cf. Figure 4.8, Section 4.2), the s-i uniform control (2.9) is the strategy with the fastest runtime on the one hand and the best performance concerning all three objectives on the other hand. Switching to a non-symmetric network (cf. Figure 4.13, Section 4.3), the uniform control strategy fails, and the slight increase in runtime of more advanced strategies is compensated by a large increase in the overall performance (at least 30%).

Conclusion. Summarizing, we observe that the advanced distribution strategies are a good tool to control production networks with random breakdowns. The qualitative behavior and the runtimes are very promising compared to the approximate optimization algorithms. In total, the best choice is the distribution strategy (2.17), (cf. Table 4.11). Additionally, we can also note that the state-independent strategies perform slightly worse than their state-dependent counterparts.

Future work might include a study for other objective functions and different network dynamics. Another open question is the application of the proposed routing strate-

networks / inflows	diamond	cascade	non-symmetric
constant	(2.12), (2.16), (2.17)	(2.13), (2.14)	(2.12), (2.14), (2.17)
stop-go	(2.12), (2.16), (2.17)	(2.12), (2.17)	(2.12), (2.17)

TABLE 4.11. *Classification of strategies, showing the best strategies for combinations of networks and inflows.*

gies to other randomly disturbed networks problems, e.g. the bounded buffer problem in [21].

REFERENCES

[1] N.T. Argon, L. Ding, K.D. Glazebrook, and S. Ziya, *Dynamic routing of customers with general delay costs in a multiserver queuing sysem*, Probability in the Engineering and Informational Sciences, 23(2), 175–203, 2009.

[2] D. Armbruster, P. Degond, and C. Ringhofer, *A Model for the Dynamics of large Queuing Networks and Supply Chains*, SIAM J. Appl. Math., 66(3), 896–920, 2006.

[3] J. Banks and J.S. Carson, *Discrete-event System Simulation*, Englewood Cliffs, New Jersey: Prentice-Hall, 1984.

[4] G. Bolch, S. Greiner, H. de Meer, and K.S. Trivedi, *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*, John Wiley & Sons, Hoboken, NJ, Second Edition, 2006.

[5] G. Bretti, C. D'Apice, R. Manzo, and B. Piccoli, *A continuum-discrete model for supply chains dynamics*, Networks and Heterogeneous Media, 2(4), 661, 2007.

[6] A. Cascone, C. D'Apice, B. Piccoli, and L. Rarità, *Circulation of car traffic in congested urban areas*, Commun. Math. Sci., 6(3), 765–784, 2008.

[7] A. Cascone, A. Marigo, B. Piccoli, and L. Rarità, *Decentralized optimal routing for packets flow on data networks*, Discrete and Continuous Dynamical Systems B, 13(1), 59–78, 2010.

[8] H. Chen and David D. Yao, *Fundamentals of Queueing Networks: Performance, Asymptotics, and Optimization(Stochastic Modelling and Applied Probability)*, Appl. Math. (New York), Springer-Verlag, New York, 46, 2001.

[9] A. Cutolo, C. D'Apice, and R. Manzo, *Traffic optimization at junctions to improve vehicular flows*, ISRN Applied Mathematics, Art. ID 679056, 19, 2011.

[10] C. D'Apice, S. Göttlich, M. Herty, and B. Piccoli, *Modeling, Simulation and Optimization of Supply Chains: A Continuous Approach*, SIAM book series on Mathematical Modeling and Computation, 2010.

[11] C. D'Apice and R. Manzo, *A fluid dynamic model for supply chains*, Networks and Heterogeneous Media, 1(3), 379–398, 2006.

[12] C. D'Apice, R. Manzo, and B. Piccoli, *Modelling supply networks with partial differential equations*, Quarterly of Applied Mathematics, 67(3), 419–440, 2009.

[13] C. D'Apice, R. Manzo, and B. Piccoli, *Existence of solutions to cauchy problems for a mixed continuum-discrete model for supply chains and networks*, J. Math. Anal. Appl., 362(2), 374–386, 2010.

[14] C. D'Apice, R. Manzo, and B. Piccoli, *Numerical schemes for the optimal input flow of a supply chain*, SIAM J. Numer. Anal., 51(5), 2634–2650, 2013.

[15] C. D'Apice and B. Piccoli, *Vertex flow models for vehicular traffic on networks*, Math. Models Meth. Appl. Sci., 18(suppl.), 1299–1315, 2008.

[16] P. Degond and C. Ringhofer, *Stochastic dynamics of long supply chains with random breakdowns*, SIAM J. Appl. Math., 68(1), 59–79, 2007.

[17] D.T. Gillespie, *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*, J. Comput. Phys., 22(4), 403–434, 1976.

[18] D.T. Gillespie, *Approximate accelerated stochastic simulation of chemically reacting systems*, The Journal of Chemical Physics, 115, 1716, 2001.

[19] S. Göttlich, M. Herty, and A. Klar, *Network models for supply chains*, Commun. Math. Sci., 3(4), 545–559, 2005.

[20] S. Göttlich, M. Herty, and A. Klar, *Modelling and optimization of supply chains on complex networks*, Commun. Math. Sci., 4(2), 315–330, 2006.

[21] S. Göttlich, S. Kühn, J.A. Schwarz, and R. Stolletz, *Approximations of time-dependent unreliable flow lines with finite buffers*, Mathematical Methods of Operations Research (MMOR), DOI: 10.1007/s00186-015-0529-6, 2016.

[22] S. Göttlich, S. Martin, and T. Sickenberger, *Time-continuous production networks with random breakdowns*, Networks and Heterogeneous Media, 6(4), 695–714, 2011.

[23] M. Gugat, M. Herty, A. Klar, and G. Leugering, *Optimal control for traffic flow networks*, Journal of Optimization Theory and Applications, 126(3), 589–616, 2005.

[24] K. Han, T.L. Friesz, and T. Yao, *A variational approach for continuous supply chain networks*, SIAM Journal on Control and Optimization, 52(1), 663–686, 2014.

[25] M. La Marca, D. Armbruster, M. Herty, and C. Ringhofer, *Control of continuum models of production systems*, IEEE Transactions on Automatic Control, 55(11), 2511–2526, 2010.

[26] A.M. Law, *Simulation Modeling and Analysis*, McGraw-Hill, Boston, 2009.

[27] N. Lee and V.G. Kulkarni, *Optimal arrival rate and service rate control of multi-server queues*, Queueing Systems. Theory and Applications, 76(1), 37–50, 2014.

[28] R. Manzo, B. Piccoli, and L. Rarità, *Optimal distribution of traffic flows in emergency cases*, European Journal of Applied Mathematics, 23(4), 515–535, 2012.

[29] Mathworks, MATLAB version 8.2.0.701 (R2013b), 2014.

[30] Y. Pochet and L.A. Wolsey, *Production Planning by Mixed Integer Programming*, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006.

[31] A.L. Stolyar, *Optimal routing in output-queued flexible server systems*, Probability in the Engineering and Informational Sciences, 19(2), 141–189, 2005.

[32] Y.-C. Teh and A.R. Ward, *Critical thresholds for dynamic routing in queueing networks*, Queueing Systems. Theory and Applications, 42(3), 297–316, 2002.

[33] S. Voß and D.L. Woodruff, *Introduction to Computational Optimization Models for Production Planning in a Supply Chain*, Springer-Verlag, Berlin, Second Edition, 2006.