COMM. MATH. SCI.
Vol. 4, No. 2, pp. 315–330

MODELLING AND OPTIMIZATION OF SUPPLY CHAINS ON COMPLEX NETWORKS*

S. GÖTTLICH†, M. HERTY‡, AND A. KLAR§

Abstract. In this paper, we extend the model for supply chains developed in [8]. The model consists of partial differential equations governing the dynamics on each processor. Furthermore, a modelling of different types of vertices is motivated and discussed. Then, optimization problems are introduced and numerically investigated. A comparison of computing times shows the efficiency of partial differential equations for solving supply chain problems.

Key words. Supply chains, conservation laws, networks, optimization

AMS subject classifications. 90B10, 65Mxx

1. Introduction

Supply chain modelling is characterized by several mathematical approaches. On the one hand there are discrete event simulations based on considerations of individual parts. On the other hand, continuous models using partial differential equations have been introduced, e.g. [2, 3, 4] and [6] for a general overview.

Here, we consider continuous supply chain models based on partial differential equations. They have been derived using a time recursion process in [2]. Therein, a supply chain of M processors is considered and each supplier m is linked to only one previous supplier $m-1$. A continuous model can be derived as follows: Let $\tau(m,n)$ denote the arrival time of part n at supplier m and let $T(m)$ denote the processing time. Moreover, let $\mu(m)$ be the maximal processing rate. Then, we obtain the time recursion as

$$\tau(m+1,n)=\max\{\tau(m,n)+T(m),\tau(m+1,n-1)+\frac{1}{\mu(m)}\}, \qquad m=0,\dots,M-1, \quad n\geq 0. \tag{1.1}$$

We supplement the recursion (1.1) with initial conditions:

$$f_1(\tau(0,n))=\frac{1}{\tau(0,n+1)-\tau(0,n)} \tag{1.2a}$$

$$\tau(m+1,0)=\tau(m,0)+T(m). \tag{1.2b}$$

The first condition (1.2a) describes the time when a part n arrives at the first processor in the chain. If the network is empty, equation (1.2b) provides the time when a part n arrives at supplier $m+1$ starting at supplier m. The recursion (1.1) is evaluated by using Newell-curves $U(m,t)$ (see [10]) defined as:

$$U(m,t)=\sum_{n=0}^{\infty} H(t-\tau(m,n)), \quad m=0,\dots,M, \quad t>0. \tag{1.3}$$

*Received: October 24, 2005; accepted (in revised version): February 24, 2006. Communicated by Pierre Degond.

†Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany (goettlich@mathematik.uni-kl.de).

‡Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany (herty@rhrk.uni-kl.de).

§Fachbereich Mathematik, Technische Universität Kaiserslautern, Germany (klar@itwm.fhg.de).

In [2] a conservation law for a function $\bar{u}(x,t)$ approximating Newell-curves $U(m,t)$ has been derived:

$$\partial_t \bar{u} - \min\{\frac{L}{T}\partial_x \bar{u}, \mu(x)\} = 0. \tag{1.4a}$$

Then, one can deduce a conservation law for the derivative $\bar{\rho} = \partial_x \bar{u}$:

$$\partial_t \bar{\rho} - \partial_x \min\{\frac{L}{T}\bar{\rho}, \mu(x)\} = 0 \tag{1.4b}$$

and inflow conditions derived from (1.2a). Here, L and T are averaged length and processing time. Due to possible discontinuities in $\bar{u}$, the solution $\bar{\rho}$ admits $\delta-$distributions. In [8] a similar model is introduced avoiding the $\delta-$distributions by adding suitable equations for queues accounting for the discontinuities in $\bar{u}$.

In the following we consider a network of supply chains as introduced in [8]. The approach there is extended to general network geometries and optimization problems for such networks are considered. The outline of the paper is as follows: Section 2 contains the network model. Section 3 gives several examples for the optimal routing of parts through such a network. In Section 4, a comparison of computation times of the PDE model of [8] with time recursion of [2] and described in (1.1)-(1.3) is presented.

2. Modelling supply networks

A supply network is a finite, connected directed graph $(\mathcal{J},\mathcal{V})$. Each arc $j \in \mathcal{J}$ corresponds to a supplier and each supplier has fixed constant processing time T_j, length L_j and maximal capacity μ_j. The supplier is modelled by a finite interval $[a_j, b_j]$. Each supplier j has an associated queue located at $x = a_j$. Suppliers are connected to each other at vertices $i \in \mathcal{V}$.

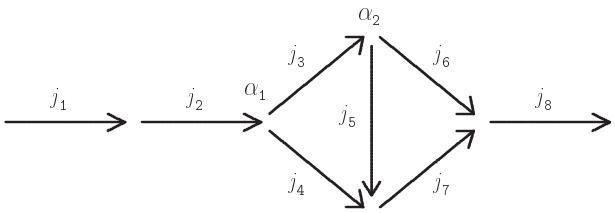

FIG. 2.1. *Sketch of a complex network with two distribution rates* α_1, α_2

We require the density (of goods or parts) ρ_j for each supplier to satisfy the advection equation with initial condition $\rho_{j,0}(x)$:

$$\partial_t \rho_j + \partial_x f_j(\rho_j) = 0, x \in [a_j, b_j], t \geq 0 \tag{2.1a}$$

$$f_j(\rho) := \min\{\frac{L_j}{T_j}\rho, \mu_j\}, \tag{2.1b}$$

$$\rho_j(x,0) = \rho_{j,0}\, x \in [a_j, b_j]. \tag{2.1c}$$

The corresponding queue is modelled as a time–dependent function $t \to q_j(t)$. The governing equation for q_j depends on the geometry of the vertex: The precise form of the equations is determined by the connected arcs at the vertex v located at $x = a_j$, see below for details.

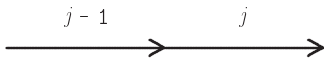

FIG. 2.2. *Connection of two suppliers*

In the simplest possible case a vertex has degree two. Let us assume a labelling and layout as in Figure 2.2; this situation has been introduced in [8].

Then, the queue q_j buffers possible demands for the processor j and hence we obtain, as in [8],

$$\partial_t q_j(t) = f_{j-1}(\rho_{j-1}(b_{j-1},t)) - f_j(\rho_j(a_j,t)), t>0. \tag{2.2}$$

Moreover, the following boundary condition has to be imposed for the outgoing supplier j at $x=a_j$:

$$f_j(\rho_j(a_j,t)) = \begin{cases} \min\{f_{j-1}(\rho_{j-1}(b_{j-1},t)),\mu_j\} & q_j(t)=0 \\ \mu_j & q_j(t)>0 \end{cases}. \tag{2.3}$$

Numerically, there is a cutoff for some $q_j<\varepsilon$ since q_j is never exactly equal to zero. For avoiding the appearing discontinuity in (2.3) we refer to [1] where a smoothed out version of (2.3) is introduced.
Summarizing, in case of a vertex of degree two the supply chain model on the outgoing arc j is given by (2.1),(2.2) and (2.3). For more details on existence of solutions in a network consisting only of vertices of degree two, we refer to [8]. There, the front tracking method is used to construct an admissible network solution and existence up to any positive time t_0 for piecewise constant initial data is proven. Next, we discuss connections at vertices of degree three. First, consider a supply chain where at a certain point the production line splits into two lines, as indicated in Figure 2.3. E.g. a good produced on arc $j-1$ might retrieve a label in the English language on arc j and in some other language on arc $j+1$.

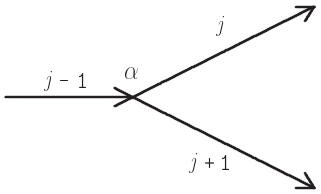

FIG. 2.3. *Splitting of a production line*

Corresponding to the idea in traffic flow (see [5] for more details), disribution rates α are introduced for controlling the flux from the previous supplier. Assume the arriving goods on arc $j-1$ continue on arc j according to a given rate α, e.g., finally 20% of all goods get a label in the English language. Obviously, we obtain the following equations for the queues q_j and q_{j+1}:

$$\partial_t q_j = \alpha f_{j-1}(\rho_{j-1}(b_{j-1},t)) - f_j(\rho_j(a_j,t)) \tag{2.4a}$$

$$f_j(\rho_j(a_j,t)) = \begin{cases} \min\{\alpha f_{j-1}(\rho_{j-1}(b_{j-1},t)), \mu_j\} & q_j(t) = 0 \\ \mu_j & q_j(t) > 0 \end{cases}, \tag{2.4b}$$

$$\partial_t q_{j+1} = (1-\alpha) f_{j-1}(\rho_{j-1}(b_{j-1},t)) - f_{j+1}(\rho_j(a_j,t)), \tag{2.4c}$$

$$f_{j+1}(\rho_{j+1}(a_{j+1},t)) = \begin{cases} \min\{(1-\alpha) f_{j-1}(\rho_{j-1}(b_{j-1},t)), \mu_{j+1}\} & q_{j+1}(t) = 0 \\ \mu_{j+1} & q_{j+1}(t) > 0. \end{cases} \tag{2.4d}$$

In general α might depend on t or additional external parameters. This case will be considered in a forthcoming publication. The opposite situation to Figure 2.3 is depicted in Figure 2.4 and describes the merger of two production lines.

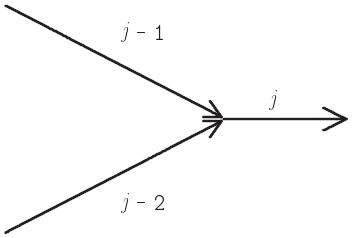

FIG. 2.4. *Merger of two production lines*

We assume that the production in process j does not need both $j-1$ and $j-2$ to be completed. Think e.g. of a filling station for soda cans. Process j fills the cans, whereas process $j-1$ and $j-2$ produces plastic and aluminium cans, respectively. In this case the governing equations for the queue q_j are:

$$\partial_t q_j(t) = f_{j-2}(\rho_{j-2}(b_{j-1},t)) + f_{j-1}(\rho_{j-1}(b_{j-1},t)) - f_j(\rho_j(a_j,t), \tag{2.5a}$$

$$f_j(\rho_j(a_j,t)) = \begin{cases} \min\{f_{j-2}(\rho_{j-2}(b_{j-1},t)) + f_{j-1}(\rho_{j-1}(b_{j-1},t)), \mu_j\} & q_j(t) = 0 \\ \mu_j & q_j(t) > 0 \end{cases}. \tag{2.5b}$$

In contrast, the situation of some processes requiring both previous suppliers (think e.g. of soda cans, where process $j-1$ delivers the can and $j-2$ delivers a label for the can), **can not** be modelled by a single queue j. Depending on the underlying production process the number of queues is given. In the case of merging cans and labels two different queues are needed. Then, the policy is to catch one can, then one label and finally put them together. If one part is missing, both queues will grow up. Further, the number of cans and labels should be the same. This concept can be adapted to other situations.

Vertices of degree higher than three (uncommon in real–world supply chains) can be modelled in a similar way as above. We omit the details.

3. Optimal routing of goods

An important question for applications is the design of 'optimal' supply networks. Depending on the actual application several questions are of importance: What is the maximal value the queues attain? Can we control the distribution such that we achieve a maximal outflow? And so forth. Here, we introduce the following general network:

Given a supply network with a vertex i of dispersing type as in Figure 2.3, suppose we can control (in some sense to be made precise below) the distribution rate α_i, then we consider:

$$\begin{gathered}\min_{\vec{\alpha}} J(\vec{\alpha},\vec{\rho},\vec{q})\\ J(\vec{\alpha},\vec{\rho},\vec{q}) := \sum_{j\in\mathcal{J}} \int_0^{T_{max}} \int_{a_j}^{b_j} \mathcal{F}(x,t,\rho_j,q_j,\vec{\alpha})\,dxdt\\ \text{subject to } \alpha_i^l \le \alpha_i \le \alpha_i^r\\ \text{and } \vec{\rho} \text{ satisfies (2.1) and for each vertex either}\\ i\in\mathcal{V}_1, j, j-1\in\mathcal{S}_i : (2.2),(2.3),\\ \text{or } i\in\mathcal{V}_2, j, j-1, j-2\in\mathcal{S}_i : (2.4),\\ \text{or } i\in\mathcal{V}_3, j, j+1, j+2\in\mathcal{S}_i : (2.5) \text{ holds.}\end{gathered} \tag{3.1}$$

Here, $\mathcal{V}=\mathcal{V}_1\cup\mathcal{V}_2\cup\mathcal{V}_3$ and $\mathcal{V}_1$ are vertices of degree two, $\mathcal{V}_2$ are vertices of dispersing type (Figure 2.3) and $\mathcal{V}_3$ are vertices of merging type (Figure 2.4). For each $i\in\mathcal{V}$ the set $\mathcal{S}_i$ is the set of all arc indices j which are connected at i. The constants $0\le\alpha_i^{l,r}\le 1$ are assumed to be known and are bounds on the minimal and maximal distribution rate. Further, $\vec{\alpha}:=(\alpha_i)_{i\in\mathcal{V}_2}$, and $\vec{\rho}:=(\rho_j)_{j\in\mathcal{J}}$. We will report on numerical results for (3.1) by applying different networks. The precise form of the functional J is given in the corresponding examples. The first three examples are sample problems whereas in Section 3.4 we deal with a real-world example of similar structure.

For solving (3.1) we apply Upwind discretizations with a step–length of $\Delta h=1/10$ for the partial differential equations for ρ_j and an explicit Euler for the ordinary differential equations for q_j. The trapezoid rule is used for discretization of the integrals appearing in the cost functional. Then, the presented computations work with the Optimization Toolbox of Matlab [9] using a routine for box constrained nonlinear problems.

REMARK 3.1. *Usually, we assume an artifical first arc with $a_1=-\infty$ and $b_1=0$ with inflow profile $f_1(t)$. For simplification, we set $L_1/T_1=1$ and $\mu_1>\max f_1$, so that the inflow profile can be translated into boundary data $\rho_2(a_2,t)=f_2^{-1}(\rho_1(b_1,t))$, see [8] for details.*

3.1. Optimization with two controls First, we present a sample optimization problem where we try to find the optimal distribution rates $0\le\alpha_i^{l,r}\le 1, i=1,2$ and $i\in\mathcal{V}_2$, see Figure 3.1. Here, α_1 of the parts coming from arc 2 continue on arc 3 and similarly α_2 coming from arc 3 continue on arc 6. We allow for different processing rates μ_j, but constant length $L_j=1$ and processing time $T_j=1$. The cost functional penalizes large values in the queues q_j, see (3.2). Initial data for the queues is given by $q_j(t)=0, j=2,\dots,8$. Then, the functional for the optimization problem (3.1) is

given by

$$\min_{\alpha_1,\alpha_2} J(\alpha_1,\alpha_2)$$
$$J(\alpha_1,\alpha_2) := \sum_{j=2}^{8} \int_0^{T_{max}} q_j^2(t)\,dt, \tag{3.2}$$

where $T_{max} = 20$.

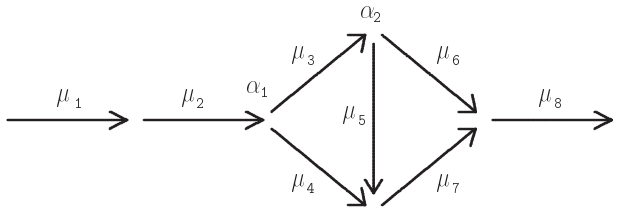

FIG. 3.1. *Network with two controls and initial maximal processing rates* $\mu_1 = 99, \mu_2 = 40, \mu_3 = 30, \mu_4 = 20, \mu_5 = 20, \mu_6 = 5, \mu_7 = 10, \mu_8 = 10$

As inflow profile we use a hat-function as in Figure 3.2. We choose $\alpha_1^0 = \alpha_2^0 =$

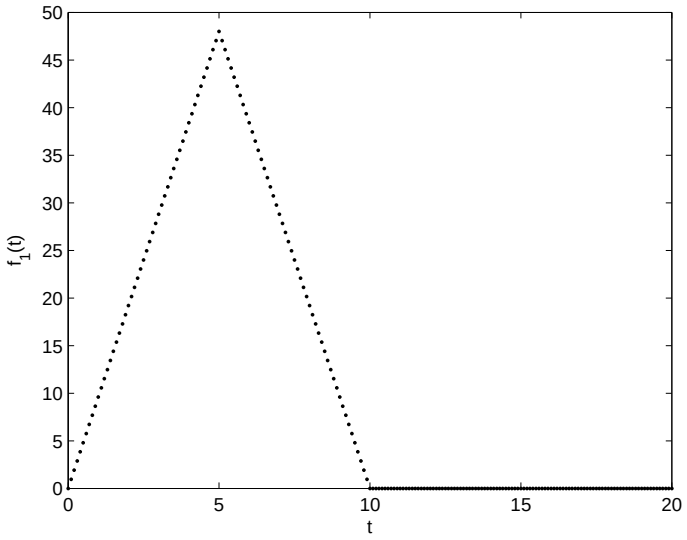

FIG. 3.2. *Inflow profile* $f_1(t)$

0.5 as an initial value for the optimization. Its corresponding functional value is $J(\alpha_1^0, \alpha_2^0) = 83002.9$. Furthermore, the naive choice $\alpha_1 = 0.5$ and $\alpha_2 = 0$ (to avoid the bottleneck on arc 6) yields a significantly higher functional value. As described in Section 3.1, a blackbox optimization code of [9] is used. After 8 iterations and 41 functional evaluations the algorithm terminates because the default tolerance 10^{-6} is reached.

The optimal distribution is $\alpha_1^{opt}=1$, $\alpha_2^{opt}=0.385097$ and $J(\alpha_1^{opt},\alpha_2^{opt})=58394.1$. In Figure 3.3 the evolution of queues for the optimal distribution is plotted. Because of $\alpha_1^{opt}=1$, the queue q_4 and also q_5 remain empty. All other queues are partly filled. The queues q_6 and q_8 remain partly filled, even so T_{max} is reached. The functional $J(\alpha_1,\alpha_2)$ is plotted in Figure 3.4. We observe that the functional $J(\alpha_1,\alpha_2)$ has a unique minimum and steep gradients for $\alpha_2\to 1$. This corresponds to the fact, that the processing rate μ_6 is the lowest in the complete network and therefore needs a large queue if filled by parts from arc 3.

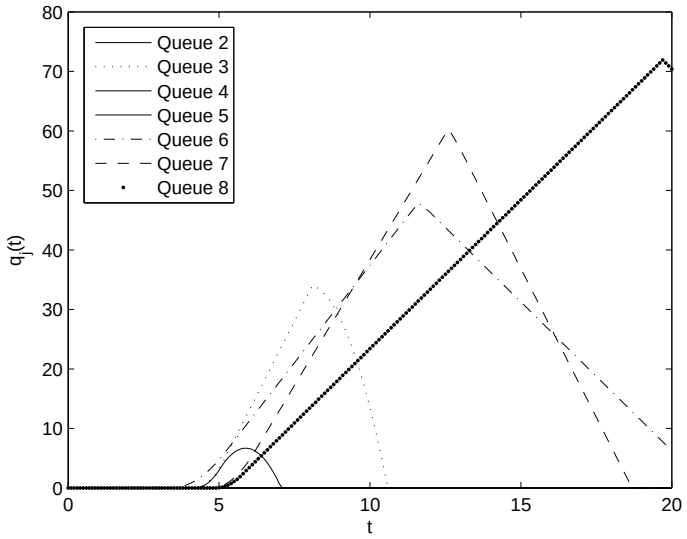

FIG. 3.3. *Amount of parts in each queue for optimal distribution rates* $\alpha_1^{opt}=1$ *and* $\alpha_2^{opt}=0.385097$. *Queue* 4 *and Queue* 5 *remain empty.*

3.2. Optimization of processing velocities. Next, we consider an optimization problem where we assume the (processing -)velocity $v_j:=L_j/T_j$ of each processor j to be variable. In a production line this corresponds to the question of optimal operating velocities for each individual processing unit j. Maximal v_j^l and minimal v_j^r processing velocities are imposed as upper and lower bounds.

For simplicity, we consider a supply network consisting of three processors $j=2,3,4$ as indicated in Figure 3.5. Arc 1 is needed to prescribe a certain demand, i.e. an inflow profile.

We assume a fixed inflow profile $t\to f_1(t)$ to be given at $j=1$ e.g., think of an output of another external process. The dynamics in each processor j is governed by (2.1). Now, the maximal processing rates μ_j are fixed and not subject to change. The controls are the ratios $v_2:=L_2/T_2$ and $v_3:=L_3/T_3$, i.e., the processing velocities of processor $j=2$ and $j=3$. Given some default velocities $L_{2,0}/T_{2,0}$ and $L_{3,0}/T_{3,0}$ we try to minimize the height of the buffering queues (as before) and producing a certain a priori prescribed outflow $f_{4,0}(t)$. To avoid triviality, we assume that $f_{4,0}(t)=f_4(\rho_4(b_4,t))$ when using the default processing velocities $L_{2,0}/T_{2,0}$ and $L_{3,0}/T_{3,0}$. To be more precise, we consider the following minimization problem.

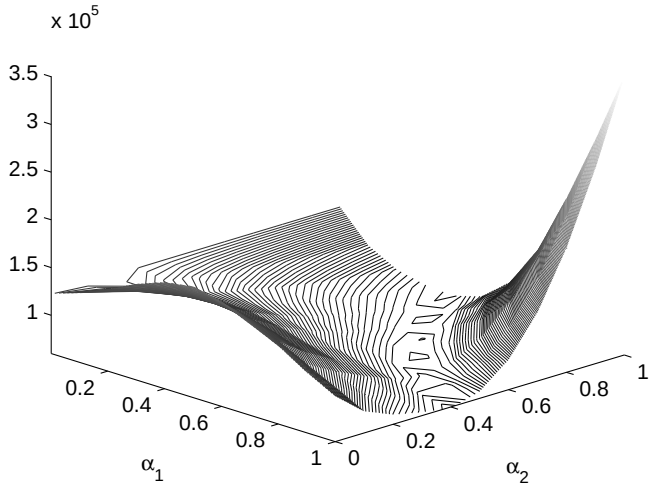

FIG. 3.4. *Plot of $J(\alpha_1,\alpha_2)$ varying α_1, α_2 in Figure 3.1*

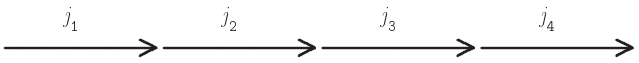

FIG. 3.5. *Network for optimization of processing velocities*

$$\min_{v_2,v_3} J(v_2,v_3)$$

$$J(v_2,v_3)=\sum_{j=2}^{4}\int_0^{T_{max}} q_j^2(t)\,dt+\int_0^{T_{max}}\left(f_4(\rho_4(b_4,t))-f_{4,0}(t)\right)^2 dt$$

$$\text{subject to } v_j^l\le v_j\le v_j^r,\ j=2,3,$$

$$\partial_t\rho_j+\partial_x\min\{\mu_j,v_j\rho\}=0,\ j=2,3,4,\ x\in[a_j,b_j],\ t\in[0,T_{max}],$$

$$\partial_t q_j(t)=f_{j-1}(\rho_{j-1}(b_{j-1},t))-f_j(\rho_j(a_j,t)),\ t\in[0,T_{max}],$$

$$q_j=0,\ j=2,3,4. \tag{3.3}$$

Here, v_j^l and v_j^r are given lower and upper bounds and $f_j(\rho_j(a_j,t))$ for $j=2,3,4$ is given by (2.3). Moreover, $f_{4,0}=f_4(\rho_4^0(x=b_4,t))$, ρ_j^0 is the solution to (2.1) for $v_2\equiv L_{2,0}/T_{2,0}$, $v_3\equiv L_{3,0}/T_{3,0}$ and $\rho_{1,0}(b_1,0)=f_1(t)$.

We present solutions to the following test case: Set $\mu_{1,2,3,4}=(99,15,10,8)$, $L_4/T_4=1$ and as default processing velocities $L_{2,0}/T_{2,0}=1$ and $L_{3,0}/T_{3,0}=3$. The time horizon for the optimization is $T_{\max}=20$. In the example, the inflow profile is

$$f_1(t)=\frac{\mu_2}{2}\left(1+\sin\left(3\pi t/T_{\max}\right)\right).$$

The associated outflow with the default velocities is then $\int_0^{T_{max}} f_{4,0}(t)dt\approx 115$ and the evolution of the queues $q_j(t)$ is plotted in Figure 3.6 (left). A point of first–

order optimality could be found after 7 iterations and 34 functional evaluations. The optimal velocities are $v_2^{opt} = 0.0255464$ and $v_3^{opt} = 1.2$. The outflow is $\int_0^{T_{max}} f(\rho_4(x = b_4, t))dt \approx 95$ and a plot of the evolution of the queues is depicted in the right part of Figure 3.6. A plot of the evolution of the fluxes is given in Figure 3.7. Obviously, and due to the slower processing speed in both processors $j = 2$ and $j = 3$, the queues q_3 and q_4 remain less used after the optimization. The maximal usage of queue two is reduced by 87% and of queue three by 29%. Moreover, this is achieved by reducing the possible outflow by only 17%.

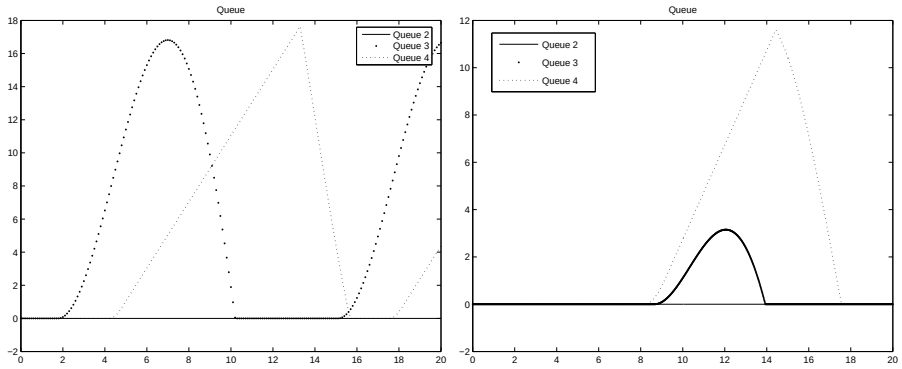

FIG. 3.6. *Evolution of the queues in test case one for the default (left) and optimal (right) velocities*

3.3. Optimization of processing velocities with bounded queues. In this section, we show that processing velocities cannot only be decelerated but also accelerated depending the cost functional. Therefore, we consider again the network given in Figure 3.5 with $\mu_{1,2,3,4} = (99, 15, 10, 8)$ and default velocities $v_j = 0.2 \forall j$. The inflow profile is constant $f_1(t) = 11$ for $0 \le t \le T_{\max} = 5$. In contrast to the simulation of the previous section, we additionally assume that the queues are bounded by some constants $q_j^{\max}$.

We measure outflow of the network and penalise queues exceeding the corresponding bound. A possibility is the following optimization problem

$$\begin{aligned} &\min_{v_2, v_3} J(v_2, v_3) \\ J(v_2, v_3) = &\max_j H^j \left(\max_t q_j(t) \right) - \int_0^{T_{max}} f_4(\rho_4(b_4, t))\, dt, \\ &\text{subject to } v_j^l \le v_j \le v_j^r,\ j = 2, 3. \end{aligned} \tag{3.4}$$

with H^j a Heaviside like function

$$H^j(q) = \begin{cases} 0 & q \le q_j^{\max} \\ 10(q - q_j^{\max}) & q > q_j^{\max}. \end{cases} \tag{3.5}$$

We consider two different scenarios. First, we consider the case of unlimited queues, i.e., $q_j^{\max} = +\infty$. In this simple setting, we expect that all velocities v_j will attain their upper bounds $v_j = v_j^r$, since this guarantees the maximal throughput. In fact, for lower and upper bounds $v_j^l = 0.2$ and $v_j^r = 1$, respectively, problem (3.4)

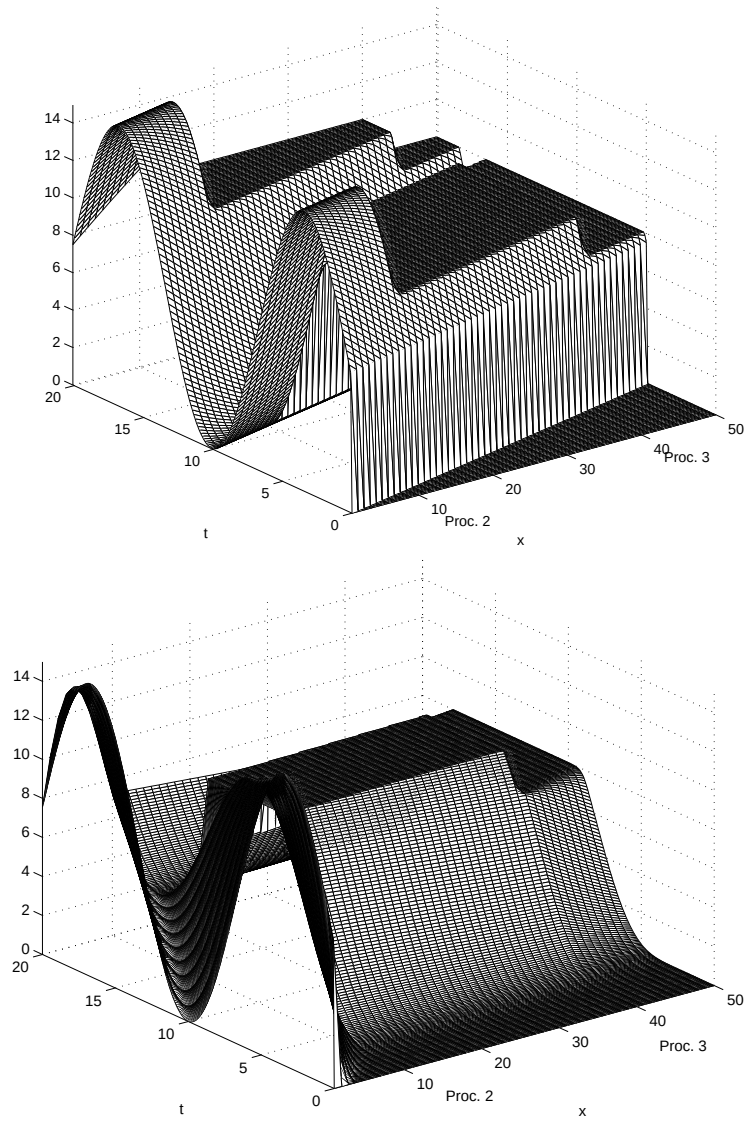

FIG. 3.7. *Evolution of the fluxes $f_j(x,t)$ in test case one for the default (top) and optimal (bottom) velocities. Processor one is located in $0 \le x \le 10$ and processor two at $10 \le x \le 40$*

is solved after 2 iterations and 11 functional evaluations with the expected result $v_{2,3}^{opt} = (1,1)$. As seen in Figure 3.8 we have a constant increase in the length of the buffering queues $q_{3,4}$.

In practical examples it is desirable to have an upper bound on the queue length. Therefore, we apply the following bounds,

$$q_{3,4}^{max} = (4,8), \tag{3.6}$$

leaving the other parameters unchanged. Then, the optimization problem is solved

after 11 iterations and 187 functional evaluations. The optimal processing velocities have changed and are given now by $v_{2,3}^{opt} = (0.2, 0.94)$. The corresponding picture of the evolution of the queues is given in the right part of Figure 3.8. Still we observe an increase in the processing velocity v_3 compared to the default initial value $v_3 = 0.2$. When comparing with the unrestricted queues we observe that the second processor works at minimal velocity to prevent the build–up of buffering queues on processor $j = 3$. Note that neither bound on q_j is reached for the optimal velocities. Obviously, the slower processing speed on $j = 2$ also effects the total throughput $\tilde{J} := \int_0^{T_{\max}} f_4 dt$. In the unrestricted case we obtain $\tilde{J} = 29.6$ compared to $\tilde{J} = 27.76$ in the restricted case.

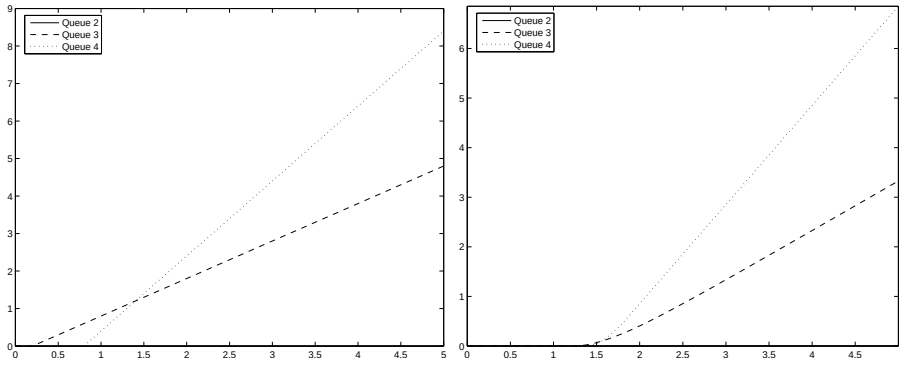

FIG. 3.8. *Evolution of queues in the case of unrestricted (left) and length restricted (right) queues.*

3.4. A real-world example. We now present a real-world example and apply some of the previous optimization ideas introduced before. The original network is illustrated in Figure 3.9 and modeled by a network with one cycle (c.f. Figure 3.10). This is an assembly line for the production of toothbrushes. These toothbrushes are fixed on paletts which are circling around in the network. In processor 1 the paletts are fed into the network and in processor 2 semifinished toothbrushes are put onto each pallet. The processors 3,4, and 5 describe some production steps. At the end of processor 5 the finished toothbrushes are taken out. The data of the suppliers is given in Table 3.1.

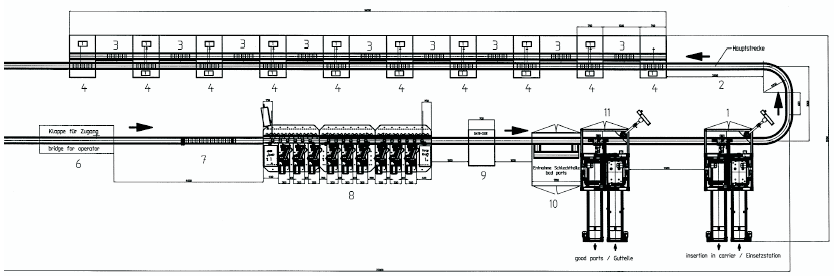

FIG. 3.9. *Original network with four suppliers connected by an assembly line.*

Processor j	μ_j	T_j	L_j	v_j
1	99	1	1	1
2	0.71	4.2	1.5	0.357143
3	0.85	70	16	0.228571
4	0.71	63	3	0.047619
5	0.53	16.8	3	0.178571

TABLE 3.1. *Maximal processing rates μ_j, processing times T_j, lengths L_j and processing velocities $v_j = \frac{L_j}{T_j}$*

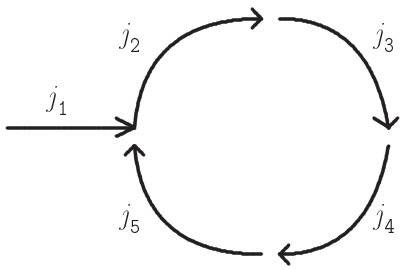

FIG. 3.10. *Network with one cycle modeling Figure 3.9*

We are interested to find possibilities avoiding the bottleneck in processor 5 having minimal μ_j. We measure the usage of the queues as in the previous section, see Section 3.1 and 3.2. The real-world data is a piecewise constant inflow profile with $T_{max} = 400$:

$$f_1(t) = \begin{cases} 1.2 \cdot \mu_2 & 0 \le t \le 150 \\ 0 & 150 < t \le T_{max} \end{cases} \tag{3.7}$$

First, we optimize the processing velocity of processor 5. We consider the following optimization problem in v_5:

$$\begin{aligned} &\min_{v_5} J(v_5) \\ J(v_5) &:= \sum_{j=2}^{5} \int_0^{T_{max}} q_j^2(t)\, dt \\ &\text{subject to } v_5^l \le v_5 \le v_5^r. \end{aligned} \tag{3.8}$$

The default velocity is $v_5^0 = 0.178571$ and $J(v_5^0) = 197442$. After 17 iterations and 54 functional counts the algorithm terminates reaching the default tolerances. The optimal processing velocity is $v_5^{opt} = 0.0120627$ and $J(v_5^{opt}) = 89500.2$. The results are given in Figure 3.11. Obviously, the amount of parts in q_2 and q_5 is reduced in the optimum and q_5 is even empty for some $t^* < T_{max}$.

Another possibility to avoid the bottleneck in processor 5 is to introduce a parallel processor. This additional processor should have the same properties as the original one (see Figure 3.12). By construction we obtain a vertex of dispersing type with control α.

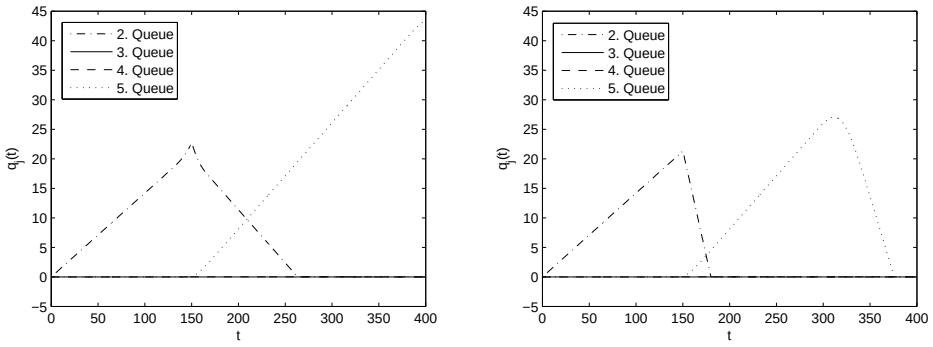

FIG. 3.11. *Amount of parts in queues $q_j(t), j=2,3,4,5$ for default (left) and optimal (right) velocity*

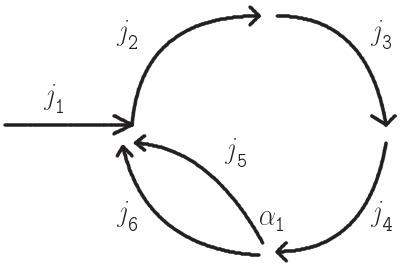

FIG. 3.12. *Network with one cycle and one parallel processor j_6*

We formulate an optimization problem as in Section 3.1:

$$\min_{\alpha_1} J(\alpha_1)$$
$$J(\alpha_1) := \sum_{j\in\mathcal{J}} \int_0^{T_{max}} \sum_{j=2}^{6} \int_0^{T_{max}} q_j^2(t)\,dt$$
$$\text{subject to } \alpha_1^l \le \alpha_1 \le \alpha_1^r. \tag{3.9}$$

It is obvious, that J is symmetric and therefore, we apply box constraints as $\alpha_1^l = 50\%$ and $\alpha_1^r = 100\%$. Initially, we start with $\alpha_1^0 = 75\%$ and $J(\alpha_1^0) = 93538.7$. Then, $\alpha_{opt} = 0.819004$ and $J(\alpha_1^{opt}) = 74945.6$ is obtained after 5 iterations and 21 functional evaluations. In Figure 3.13 on the left the queues are plotted for $\alpha_1 = 1$ and on the right the evolution of the queues for α_{opt} is shown.

4. Comparison of CPU Times

We give results on the computational effort for solving the introduced PDE model (2.1) and compare with the discrete model (1.1)-(1.3). We consider the network shown in Figure 3.5, i.e. we consider three, respectively, six suppliers with different properties L_j, T_j, μ_j and a constant inflow profile $f_1 = const$, such that the amount of parts n for the discrete event simulation (1.1) is:

$$\int_0^{T_{max}} f_1(t)\,dt = n. \tag{4.1}$$

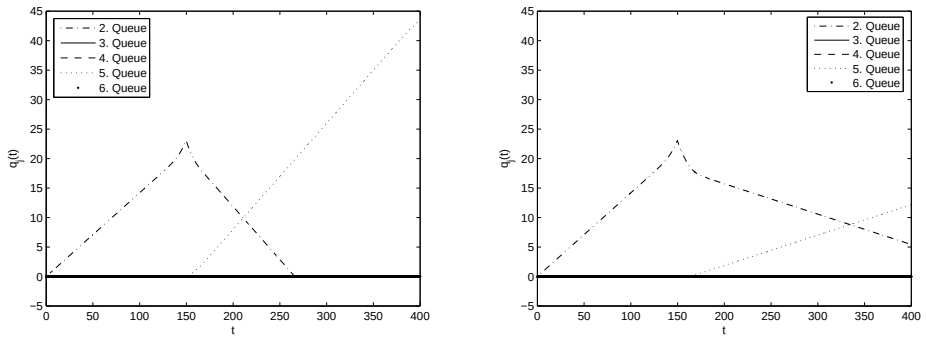

FIG. 3.13. *Evolution of queues for the initial situation with* $\alpha_1 = 1$ *(left) and in the optimimum* $\alpha_1 = 0.819004$ *(right)*

In the following we call (1.1) **D**iscrete **E**vent **S**imulation (DES). In contrast to the DES the computational effort for solving the PDE model given by (2.1) is dependent on time and space discretizations. Again, we use the discretization presented in Section 3. In Table 4.1 and 4.2 we present different time discretizations. We report three times for the DES (1.1)-(1.3):

- First, the computation of initial conditions (1.2a) with linear interpolation and the computation of (1.2b).
- Second, the time used to compute the matrix $A_{m,n} = (\tau(m,n))$ in (1.1).
- Third, the evaluation of (1.1) by Newell-curves (1.3).

In Table 4.1, we observe that solving the PDE is comparable with computing 10000 parts for the DES. Further, we recognize that the CPU times grow linear with the number of parts in the discrete model. In Table 4.2 we present results for finer discretization of the PDE model. Now, the CPU time used to compute the PDE model corresponds to computing 50000 parts.

Model	Parameters		CPU Time [sec]			
			(1.2a) + (1.2b)	(1.1)	(1.3)	Total
DES	$K=3$ $T_{max}=100$ $\Delta t=0.5$	$n=10000$	2.0229	0.010014	0.91331	2.9442
DES		$n=50000$	9.6238	0.020029	4.6267	14.2705
DES		$n=100000$	19.0374	0.040058	9.4937	28.5711
PDE		$\Delta x=0.1$				1.4220
DES	$K=3$ $T_{max}=1000$ $\Delta t=0.5$	$n=10000$	3.1646	0.010014	7.9514	11.126
DES		$n=50000$	15.1718	0.020029	44.6542	59.8461
DES		$n=100000$	30.0132	0.040058	92.8635	122.9167
PDE		$\Delta x=0.1$				13.6396
DES	$K=6$ $T_{max}=1000$ $\Delta t=0.5$	$n=10000$	3.0844	0.010014	14.8714	17.9658
DES		$n=50000$	15.0917	0.040058	88.5573	103.6891
DES		$n=100000$	30.0132	0.080115	180.5696	210.6629
PDE		$\Delta x=0.1$				27.8701

TABLE 4.1. *CPU Times for coarse discretisation* $\Delta t = 0.5$

Model	Parameters		CPU Time [sec]			
			(1.2a) + (1.2b)	(1.1)	(1.3)	Total
DES	$K=3$	$n=10000$	2.0329	0.010014	4.5365	6.5795
DES		$n=50000$	9.5838	0.020029	22.963	32.5668
DES	$T_{max}=100$	$n=100000$	19.0574	0.040058	47.0779	66.1752
PDE	$\Delta t=0.1$	$\Delta x=0.02$				34.79
DES	$K=3$	$n=10000$	3.1545	0.010014	40.8487	44.0033
DES		$n=50000$	15.0917	0.020029	223.7918	238.7918
DES	$T_{max}=1000$	$n=100000$	30.0732	0.040058	463.8971	494.0104
PDE	$\Delta t=0.1$	$\Delta x=0.02$				346.6284
DES	$K=6$	$n=10000$	3.3749	0.010014	75.3383	78.7232
DES		$n=50000$	15.2019	0.030043	446.7624	461.9943
DES	$T_{max}=1000$	$n=100000$	30.1233	0.080115	902.8582	933.0617
PDE	$\Delta t=0.1$	$\Delta x=0.02$				697.0623

TABLE 4.2. *CPU Times for fine discretisation* $\Delta t=0.1$

5. Summary and Conclusions

In the present paper, we defined coupling conditions at vertices of complex networks and extended the work of [8]. Furthermore, suitable optimization problems are introduced for a variety of examples. We give numerically results on sample and real-world problems as well as a comparison of the PDE model and the already known discrete event model.

Future work will include improved optimization procedures as for example an adjoint and sensitivity calculus. Moreover, the present work can be extended to obtain simplified models as for example proposed in [7]. This is connected to the exploration of connections to the description of supply chains by methods of discrete optimization like Mixed-Integer Problems (MIP). In this context also time dependent controls will be included in a straightforward way.

Acknowledgements. This work was supported by the University of Kaiserslautern Excellence Cluster 'Dependable Adaptive Systems and Mathematical Modeling'.

REFERENCES

[1] D. Armbruster, C. de Beer, M. Freitag, T. Jagalski and C. Ringhofer, *Autonomous control of production networks using a pheromone approach*, Physica, submitted, 2005.
[2] D. Armbruster, P. Degond and C. Ringhofer, *A model for the dynamics of large queuing networks and supply chains*, SIAM J. Applied Mathematics, submitted, 2004.
[3] D. Armbruster, P. Degond and C. Ringhofer, *Kinetic and fluid models for supply chains supporting policy attributes*, Transp. Theory and Stat. Phys., submitted, 2004.
[4] D. Armbruster, D. Marthaler and C. Ringhofer, *Kinetic and fluid model hierarchies for supply chains*, SIAM J. Multiscale Modeling and Simulation, 2(1), 43-61, 2004.
[5] G. Coclite, M. Garavello and B. Piccoli, *Traffic flow on road networks*, SIAM J. Math. Anal., 36, 1862-1886, 2005.
[6] C. F. Daganzo, *A Theory of Supply Chains*, Springer Verlag, New York, Berlin, Heidelberg, 2003.
[7] A. Fügenschuh, M. Herty, A. Klar and A. Martin, *Combinatorial and Continuous Models for the Optimization of Traffic Flows on Networks*, SIOPT, to appear.

[8] S. Göttlich, M. Herty and A. Klar, *Network models for supply chains*, Comm. Math. Sci., 3(4), 545-559, 2005.
[9] Matlab Version 7.0
http://www.mathworks.com/access/helpdesk/help/toolbox/optim/optim.shtml
[10] G. F. Newell, *A simplified theory of kinematic waves in highway traffic*, Transportation Research, 27B, 281-313, 1993.