Optimization of order policies in supply networks

S. Göttlich* M. Herty† C. Ringhofer‡

August 18, 2008

Abstract

The purpose of this paper is to develop a model which allows for the study and optimization of arbitrarily complex supply networks, including order policies and money flows. We propose a mathematical description that captures the dynamic behavior of the system by a coupled system of ordinary differential delay equations. The underlying optimization problem is solved using discretization techniques yielding a mixed–integer programming problem.

KEYWORDS: Policy models, production lines, optimization
AMS CLASSIFICATION: 65N35, 65N05

1 Introduction

This paper is concerned with the modeling and optimization of supply networks. We consider a network of suppliers, each processing a product at various stages. Each supplier receives orders for its output from the other suppliers in the network, as well as from a final customer. Each supplier orders its input from a select set of other suppliers in the network as well as from a raw material supplier. Each supplier receives payments for delivered items at a certain fixed set of prices and has to pay production costs at certain rates for each item produced. This setup is similar to the ones studied in [4], [5] and [7], where it is shown that the resulting dynamics can exhibit a quite complicated behavior. One of the key features of the model

*TU Kaiserslautern, Department of Mathematics, Postfach 3049, 67653 Kaiserslautern, Germany (goettlich@mathematik.uni-kl.de).

†RWTH Aachen, Department of Mathematics, Templergraben 55, 52056 Aachen, Germany (herty@mathc.rwth-aachen.de.).

‡Arizona State University, Department of Mathematics, Tempe, AZ 85287-1804, USA (ringhofer@asu.edu).

considered in this paper is the presence of limited production capacities. We assume that each supplier has a certain limited capacity μ, i.e. it cannot process more than $\mu\Delta t$ items in the infinitesimal time interval Δt. We use a generalization of the model employed in [1] to incorporate this additional limitation. The inclusion of payments in the model allows for the definition of capital flows through the network and, consequently, for the occurrence of bankruptcies if the capitalization of a node in the network falls below a certain threshold.

One of the key components of the work in this paper is the optimization of profits, i.e., given a set of production costs, to choose ordering strategies to maximize the income of each node. The optimization of routing strategies for networks of almost arbitrary complexity has been investigated in [11] using an adjoint calculus approach. In this paper, we use a different approach, namely using a mixed integer program (MIP) [8] , [15]. To develop an understanding of the structure of possible optimal solutions, all possible steady states are first characterized and investigated analytically.

The rest of this paper is organized as follows: Since the resulting model is relatively involved, we use Section 2 to explain the features and concepts of the model in detail. Since the final goal is to construct steady states which are in some sense optimal, i.e. which optimize a certain functional, we characterize possible steady states of the model in Section 3. The formulation of the resulting optimization problem as a mixed integer program is not completely straight forward. So, Section 4 is devoted to setting up the mixed integer program. Section 5 is devoted to numerical experiments. We test the optimization approach on two different sample problems. The first scenario shows the strong connection betweeen money flows and order policies with the objecive of maximizing the profit of each individual supplier. However, the second example describes how bankruptcy and production failures influence orders in highly interconnected networks.

2 The model

2.1 The supplier model

The basic setup in this paper requires two inventories for each supplier, namely

1. An input inventory, allowing for the storage of items the supplier has ordered at a rate faster than it can process. (This might be a reasonable strategy if the supplier expects shortages in the future.)

2. An output inventory, allowing for the storage of items which have been produced at a rate faster than ordered. (This might be a reasonable strategy if the supplier anticipates a future demand larger than its capacity.)

A simple deterministic inventory model:
We define a simple deterministic model for inventories. The inventory has an influx ψ and an output μ. Thus the inventory position p would satisfy the simple differential equation $p'(t) = \psi - \mu$. On the other hand, the inventory position cannot become negative. Thus, in addition to the simple ODE, we also have to satisfy the constraint $p \geq 0$, and are therefore faced with an obstacle problem. We use a relaxation model for the obstacle problem, replacing the differential equation by

$$\frac{dp}{dt} = \max\{\psi - \mu, -\frac{p}{\varepsilon}\} \ , \tag{1}$$

where ε denotes a small relaxation parameter. Equation (1) has the following features:

- As long as the inventory is growing ($\psi - \mu > 0$) the inventory in the model (1) grows according to $\frac{dp}{dt} = \psi - \mu$.
- If the inventory position is decreasing ($\psi - \mu < 0$), the inventory in (1) decays until $p = \varepsilon(\mu - \psi)$ holds, i.e. until the inventory is almost empty. From this point on equation (5) becomes $\frac{dp}{dt} = -\frac{p}{\varepsilon}$, and the inventory position decays exponentially to zero on an $O(\frac{1}{\varepsilon})$ time scale, until the net flux $\psi - \mu$ becomes positive again, and we switch to the previous regime.

Thus, the inventory position $p(t)$ in (1) will remain always positive, and represent an $O(\varepsilon)$ approximation of the (deterministic) evolution of an actual inventory. Equation (1) represents an approximate relaxation model where the actual depletion of the inventory in finite time is replaced by an exponential decay to zero.

As mentioned above, we need two inventories to describe the evolution of a single node, namely an input and an output inventory. We consider a network of K nodes. The input inventory p_k of node number $k = 1 : K$ receives an influx ψ_k, and has an outflux, given by the processing rate μ_k. Thus we have

$$\frac{dp_k}{dt} = \max\{\psi_k - \mu_k, -\frac{p_k}{\varepsilon}\} = \psi_k - \phi_k, \quad \phi_k = \min\{\mu_k, \frac{p_k}{\varepsilon} + \psi_k\} \ . \tag{2}$$

The term ϕ_k denotes the flux from the inventory into the processor. Obviously $\phi_k \leq \mu_k$ holds, since the processor cannot accept items at a rate larger than μ_k. Assuming a (deterministic) processing time τ_k, the influx into the output inventory is given by $\phi_k(t - \tau_k)$, i.e. the time delayed outflux of the input inventory. The evolution of the output inventory position q_k is then given in the same way by

$$\frac{dq_k}{dt} = \max\{\phi_k(t - \tau_k) - \omega_k, -\frac{q_k}{\varepsilon}\} = \phi_k(t - \tau_k) - f_k, \tag{3}$$

$$f_k = \min\{\omega_k, \frac{q_k}{\varepsilon} + \phi_k(t - \tau_k)\} \ ,$$

with ω_k the rate of orders received by processor k, and f_k the total outflux of node number k. Again, f_k cannot exceed ω_k, and node number k cannot deliver at a faster rate than orders are received. The inventory model used here essentially corresponds to replacing the inventory by a processor with a very short cycle time ε, and then using a version of the model for a production unit developed in [2] and [10]. Similar simple inventory models have been used in [1] and [3] in a different context.

2.2 The network model

Given the evolution of the input and output inventories, defined in Section 2.1, we now have to define the interaction of the different nodes in the supply network. This means, we have to define the influx ψ_k of node number k in (2) in terms of the outfluxes f_k of the other nodes, given by (3), and we have to decide on a rule for the order rates ω_k in (3). One of the key mechanisms, governing the dynamics of the system is obviously the policy of placing orders. In general, we will denote with Ω_{jk} the rate at which node number j places orders to node number k, and the total rate of orders received by node number k in (3) is given by $\omega_k = \sum_j \Omega_{jk}$.

The second mechanism, governing the dynamics, is a distribution policy. The need for this policy arises in the case that not all orders can be filled. We define by F_{jk} the flux from node k into node j. Consequently, $f_k = \sum_j F_{jk}$ is the total outflux of node number k. Because of (3) we already have that $f_k \leq \omega_k$ holds. The need for a distribution policy arises when f_k is actually strictly less than ω_k, i.e. node number k cannot satisfy all its orders and has to make a decision how to distribute its limited resources. We write the flux from node number k to node number j as $F_{jk} = A_{jk} f_k$, where the matrix $\mathbf{A} = \{A_{jk}\}$ is a Markov matrix, i.e. a matrix with non - negative entries whose column sums equal unity. For an admissible distribution policy, the matrix $\mathbf{A}$ should satisfy the following two criteria:

- $F_{jk} = A_{jk} f_k \leq \Omega_{jk}$, i.e. node k cannot deliver more to node j than node j is ordering from node k.
- If $f_k = \omega_k$ holds in (3), then $F_{jk} = \Omega_{jk}$ should hold, i.e. if node number k can satisfy all its orders, then it will do so.

In addition, we have to define an external supplier of raw materials and a final customer. We assume a single raw material supplier, defined as node number $k = 0$, and a single customer, defined as node number $k = K + 1$. So Ω_{j0} denote the rates at which raw materials are ordered, $\Omega_{K+1,k}$ denote the rates at which the final customer orders, and $F_{K+1,k}$ are the rates at which product is delivered to the customer.

There are many different ways to define a distribution policy, i.e. a matrix $\mathbf{A}$, satisfying the two criteria above. In this paper, we will restrict ourselves to the simplest one, namely a proportional policy, setting

$$A_{jk} = \frac{\Omega_{jk}}{\omega_k},\ j = 1 : K+1,\ k = 1 : K, \quad \omega_k = \sum_{j=1}^{K+1} \Omega_{jk},\ k = 1 : K\ . \tag{4}$$

The distribution policy (4) satisfies the admissibility criteria since, by definition, $F_{jk} = \frac{\Omega_{jk} f_k}{\omega_k}$ holds. Thus the inequality is always satisfied because of $f_k \leq \omega_k$, and equality holds in the case $f_k = \omega_k$. The distribution policy (4) means that, if not all orders can be satisfied, node number k distributes the product proportionally according to the orders received.

In summary, the dynamics of the flow of the network is given by equations (2) and (3) for all processor nodes $k = 1 : K$, where the influx functions ψ_k in (2) are determined by the outflux rates f_k in (3) through the connectivity matrix $\mathbf{A} = \{A_{jk}\}$. So

$$\psi_j = \sum_{k=1}^{K} A_{jk} f_k + \Omega_{j0},\ j = 1 : K \tag{5}$$

holds. Ω_{j0} denote the external inputs into the system, from the raw material supplier which, assuming an unlimited supply, equal the orders placed to the raw material supplier. After choosing an order matrix $\mathbf{\Omega} = \{\Omega_{jk},\ j = 1 : K+1,\ k = 0 : K\}$ the dynamics of the system are therefore completely defined.

The recycling trick

The system (2)-(5) represents an open system, i.e. mass is not conserved,

due to the external influx and outflux at the supplier and the customer node. For analytical purposes, it will be convenient to replace the open system by a closed system by introducing an artificial 'recycling step'. That is we artificially identify the raw material supplier with the customer, and feed the delivered product back into the system as raw material. So, Ω_{j0} still denote the order rates from the raw material supplier and $\Omega_{0k} = \Omega_{K+1,k}$ denote now the rates at which the customer orders. With this change in notation, (4) and (5) become

$$(a)\ A_{jk} = \frac{\Omega_{jk}}{\omega_k},\ j = 0:K,\ k = 0:K, \quad \omega_k = \sum_{j=0}^{K} \Omega_{jk},\ k = 0:K\ , \qquad (6)$$

$$(b)\ \psi_j = \sum_{k=0}^{K} A_{jk} f_k,\ j = 0:K\ .$$

ψ_0, the influx into the raw material supplier / customer is now the rate at which final product is delivered. The advantage of this notational trick is, that it allows for a uniform treatment of all the nodes and yields a mass conserving system. In order not to change the dynamics of the system we have to design node number $k = 0$ in such a way, that there is a limitless supply. This is easily done by giving the raw material supplier / customer formally an infinite production capacity and a zero processing time, and by making its output inventory large enough at the beginning such, that it never runs dry, i.e. we set formally $\mu_0 = \infty,\ \tau_0 = 0$ in (2) and make $q_0(0)$ sufficiently large. This has the effect that all the product delivered to the final customer immediately goes to the output inventory, and the system is fed from this (sufficiently large) output inventory q_0, i.e. we have $\phi_0 = \psi_0,\ f_0 = \omega_0$ in (2). Alternatively, we could simply change the definition of the fluxes in (2) and (3) for the node $k = 0$ to

$$\phi_0 = \psi_0, \quad f_0 = \omega_0\ ,$$

allowing the output inventory q_0 to become negative. The system is now closed and the total product is conserved, since the columns of the square matrix $\mathbf{A} = \{A_{jk},\ j,k = 0:K\}$ all add up to unity. If we define the contents of the processor number k at time t by $r_k(t)$ its evolution is given according to (2)-(3) by $\frac{dr_k}{dt} = \phi_k(t) - \phi_k(t - \tau_k)$, and the evolution of the total mass in the system is given by

$$\frac{d}{dt} \sum_{k=0}^{K} (p_k + q_k + r_k) = \sum_{k=0}^{K} \psi_k - f_k = 0\ .$$

We note that this modification of the system is done only for technical convenience, and that, if the initial supply inventory $q_0(0)$ is chosen large enough, the dynamics of the network remain unchanged.

2.3 Capital flows

In order to study the actual profitability of a given policy, it is necessary to model the flow of capital through the network. We assume that each node in the network charges a unit price $\beta_k,\ k = 0 : K$ per item delivered, and has a unit production cost $\mathfrak{c}_k$ which is paid as soon as the item leaves the input inventory and enters the processor. At the same time each node reaps a certain profit $\mathfrak{p}_k$ per unit delivered, which is taken out of the capital. In the absence of credit, i.e. each item is paid for instantaneously as it is delivered, the flow of capital through the network is therefore given by the reverse of the product flow, weighted by the price. So, the flow of capital from node number j to node number k is given by $F_{jk}\beta_k$. The evolution of the amount of capital $\kappa_k(t)$ of node number k is therefore given by the equation

$$\frac{d\kappa_k}{dt} = \sum_{j=0}^{K} F_{jk}\beta_k - F_{kj}\beta_j - \mathfrak{c}_k\phi_k - \mathfrak{p}_k f_k = \tag{7}$$

$$f_k\beta_k - \sum_{j=0}^{K} A_{kj} f_j \beta_j - \mathfrak{c}_k\phi_k - \mathfrak{p}_k f_k,\ k = 0 : K\ .$$

Notice, that the amount of capital in the system is not conserved because of the production costs and profits, given by the sink terms $\mathfrak{c}_\mathfrak{k}\phi_k$ and $\mathfrak{p}_\mathfrak{k} f_k$ in (7). The introduction of the taken profits $\mathfrak{p}_k$ allows for the definition of a steady state, which actually produces some income for the node. The role played by production costs $\mathfrak{c}_k$ and the taken profits $\mathfrak{p}_k$ is actually very similar, except that production costs are paid at production - and are therefore proportional to ϕ_k, whereas profits are realized at delivery - and are therefore proportional to f_k. Because of the recycling trick in Section 2.2, production costs and profits for node number $k = 0$, the supplier / customer are an artificial quantity, and might as well beset to zero.

We note also, that the computation of capital flows is a post - processing step, as long as the order policies are not influenced by the capitalization of each supplier. A feedback between available capital and the dynamics of the network is given by a bankruptcy mechanism (as studied in a more specific network in [5] and [7]), i.e. when the rate of orders supplier number k can place, given by $\gamma_k = \sum_{j=0}^{K} A_{kj}\omega_j$, is forced to zero as soon its capital

κ_k falls below a certain threshold. The effects of such a mechanism will be studied in Section 5.

3 Steady states

In this section, we investigate the possible steady states of the system, defined in Section 2. More precisely, given a set of capacities, production costs and a certain topography, we characterize the connection between orders prices and fluxes by essentially parameterizing the possible steady states of the system. The way we proceed is to characterize the system via its fluxes. That is we prescribe the distribution matrix $\mathbf{A}$ in (5) first, compute all possible steady fluxes for a given $\mathbf{A}$, and then find all possible time - independent order and pricing strategies which would produces the given steady state. This represents in a certain sense a backward analysis, since, in reality, the orders will determine the fluxes and not vice versa. This form of backward analysis provides, however, a useful tool to develop an analytical feeling for possible optimal solutions. The corresponding forward analysis - computing the fluxes from the orders, and trying to choose an optimal set of orders - will be carried out on a numerical level in Sections 4 and 5.

3.1 Steady state fluxes and orders

Obviously a steady state of the system is characterized, via Kirchhoff's law, by the fact that the influx into each node equals the outflux. More precisely, for p_k, r_k, q_k in (2)-(3) to be time independent, the condition $\psi_k = \phi_k = f_k$ has to hold for time independent functions ϕ_k. The condition $\psi_k = f_k$ implies, that the fluxes f_k are eigenvectors of the connectivity matrix $\mathbf{A}$, satisfying

$$\mathbf{A}f = f \ .$$

Because we have formulated the system as a closed system, the connectivity matrix $\mathbf{A}$ is a Markov matrix, i.e. a matrix with nonnegative entries whose column sums equal unity. In the following, we employ some standard properties of Markov matrices. We start with the following

Assumption 1 *The matrix $\mathbf{A} = \{A_{jk},\ j,k = 0 : K\}$ is primitive, that is, there exists an index n such that all the values of $\mathbf{A}^n$ are strictly positive.*

Assumption 1 essentially excludes only trivial cases. It basically states that a part starting out in node number k at time $t = 0$ has a nonzero probability

to appear in all the other nodes after a sufficiently long time, i.e. after iterating the matrix **A** sufficiently often. Given the recycling trick employed in Section 2, this will be the case, since the part will eventually end up at the final customer and re - enter the system as raw material. The only possibility that Assumption 1 is violated arises for the (pathological) case of closed loops, i.e. the existence of a self contained sub - graph which is not connected to the rest of the network and the raw material supplier. As a consequence, we have

Lemma 3.1 *The matrix* **A***, defined in (6) has an eigenvalue* λ_{max} *equal to unity. All other eigenvalues are strictly less than 1. The eigenspace to the eigenvalue* $\lambda_{max} = 1$ *is one dimensional. The unique normalized eigenvector* z *to the eigenvalue* $\lambda_{max} = 1$ *has only nonnegative elements.*

Proof: The first statements - that there is a single eigenvector to the eigenvalue $\lambda_{max} = 1$, and that all other eigenvalues are strictly less then $\lambda_{max} = 1$ are well known features of primitive Markov matrices. We refer the reader to c.f. [14]. Therefore, the eigenvector z can be computed via the vector iteration $y_{n+1} = \mathbf{A}y_n$, which will converge to the projection of the initial vector y_0 onto the eigenvector z. Starting with an initial vector y_0 with nonnegative elements, all iterates, and therefore also the limit z will only have nonnegative components.

■

So, for any given primitive Markov matrix **A** we can find an admissible flux vector $\psi = f$ with nonnegative entries, yielding a steady state of the system. We now proceed to characterize this vector. Given the flux definitions (2)-(3), we have

$$\psi_k = \phi_k = f_k \Rightarrow f_k = \min\{\mu_k, \frac{p_k}{\varepsilon} + f_k\} = \min\{\omega_k, \frac{q_k}{\varepsilon} + f_k\}$$

or

$$\min\{\mu_k - f_k, \frac{p_k}{\varepsilon}\} = \min\{\omega_k - f_k, \frac{q_k}{\varepsilon}\} = 0$$

This allows for 4 possible cases, namely

Case 1: $f_k < \mu_k$, $f_k < \omega_k$ and $p_k = q_k = 0$. The node runs below capacity and cannot satisfy its orders because of insufficient influx. Consequently, the inventories are empty.

Case 2: $f_k = \mu_k < \omega_k$ and $q_k = 0$ and $p_k \geq 0$. The node runs at capacity. However the orders exceed the capacity. Consequently, the output inventory is empty and the size of the input inventory in steady state is arbitrary, i.e. given by the history of the dynamics and the initial condition.

Case 3: $f_k = \omega_k < \mu_k$ and $p_k = 0$ and $q_k \geq 0$. The node satisfies its orders running below capacity. Consequently the input inventory is empty and the output inventory in steady state is arbitrary.

Case 4: $f_k = \omega_k = \mu_k$ and $p_k \geq 0,\ q_k \geq 0$. The orders precisely equal the capacity. With the correct influx f_k the position of the input and output inventories is arbitrary.

This gives rise to the following Lemma, characterizing the possible steady states of the system in terms of a scalar Ω and a vector α:

Lemma 3.2 *Given a primitive distribution matrix* $\mathbf{A}$ *and, correspondingly, a unique normalized eigenvector* z *with* $\mathbf{A}z = z$*, all possible steady states of the system are given by choosing a constant* $\gamma \leq \min\{\frac{\mu_k}{z_k},\ k = 0 : K\}$ *and a vector* $\alpha = (\alpha_0, .., \alpha_K)$ *with* $0 < \alpha_k \leq 1,\ k = 0 : K$*, and setting*

$$f_k = \gamma z_k, \quad \omega_k = \frac{\gamma z_k}{\alpha_k}, \quad \Omega_{jk} = A_{jk}\omega_k \ .$$

All orders are satisfied in steady state if $\alpha = (1, .., 1)$ *is chosen, and the system runs at its optimal capacity if* $\gamma = \min\{\frac{\mu_k}{z_k},\ k = 0 : K\}$ *holds.*

Proof: In all possible cases we have the inequalities $f_k \leq \mu_k$ and $f_k \leq \omega_k$ for each node $k = 0 : K$. The flux vector f has to be a multiple of the normalized eigenvector z. So $f_k = \gamma z_k$ has to hold, with z the normalized eigenvector to the eigenvalue problem. Since $\gamma z_k \leq \mu_k,\ k = 0 : K$ has to hold, we have $\gamma \leq \min_{k=0:K}\{\frac{\mu_k}{z_k}\}$.

For a given value of γ, we parameterize the solutions to the second set of inequalities by setting $f_k = \gamma z_k = \alpha_k \omega_k$ with $0 < \alpha_k \leq 1$. For a given vector α, this determines ω_k, the total rate of received orders. Given the proportional distribution policy, the order matrix $\mathbf{\Omega}$ is given by $\mathbf{\Omega} = \mathbf{A} \cdot diag(\omega)$.

For each index k for which $\alpha_k < 1$ holds, we do not satisfy the orders in steady state, and if $\gamma < \min\{\frac{\mu_k}{z_k},\ k = 0 : K\}$ the system is running below its bottleneck capacity.

∎

Lemma 3.2 describes how to choose the optimal orders received, the quantities ω_k given a distribution policy matrix $\mathbf{A}$. In this sub-optimal solution, the nodes with index k, for which $f_k = \gamma z_k < \mu_k$ holds, will run below capacity. To construct an optimal solution, where all nodes run at their capacity, would involve the construction of a matrix $\mathbf{A}$ such that the eigenvector z is parallel to the capacity vector μ. In this case, setting $\alpha = (1, .., 1)$, we would obtain $f_k = \mu_k = \omega_k,\ k = 0 : K$. So, all nodes would run at their capacity and all orders would be satisfied. Given, that the network will in general not allow a supplier to order from any other supplier, because not all suppliers are equal, and there is some hierarchical structure in the production process, finding the optimal matrix $\mathbf{A}$ will be impossible in general.

3.2 Steady state money flows

Computing steady states in the capital flow is complicated by the fact that the evolution of the capital, as given by (7) is not conservative. We repeat the 'recycling trick' of Section 2.2 for capital flows by assuming that production costs as well as profits incurred by the actual nodes $k = 1 : K$ are deposited at node $k = 0$, the virtual supplier / customer node. We therefore replace (7) by

$$\frac{d\kappa_k}{dt} = f_k\beta_k - \sum_j A_{kj} f_j \beta_j - \mathfrak{c}_k\phi_k - \mathfrak{p}_k f_k + \delta_{k0} \sum_j \mathfrak{c}_j\phi_j + \mathfrak{p}_j f_j \ , \qquad (8)$$

i.e. the profits $\mathfrak{p}_k f_k$ and the production costs $\mathfrak{c}_k\phi_k$ are virtually paid to the node $k = 0$. The capital κ_0 of the supplier / customer node is an artificial quantity anyway, and therefore the definition (8) does not change the dynamics of the actual flow, The advantage of the formulation (8) is, of course, that the capital flows now also form a conservative system, i.e. the quantity $\sum_{k=0}^{K} \kappa_k$ remains constant in time. This allows us to formulate the following question:

Given a certain set of production costs $\mathfrak{c}_k$ and a certain amount of desired profits $\mathfrak{p}_k$ for each node, is there a set of unit prices β_k to achieve a steady state? The answer to this question is given in the following

Lemma 3.3 *For any primitive distribution matrix* $\mathbf{A}$*, and for any set of production costs* $\mathfrak{c}_k,\ k = 1 : K$ *and profits* $\mathfrak{p}_k,\ k = 1 : K$ *there exists a pricing strategy* $\beta_k,\ k = 0 : K$ *such that the steady state product flow, given in Lemma 3.2, produces steady state capitals* $\kappa_k, k = 0 : K$*, as defined in (8).*

Proof: Given a steady state, as defined in Lemma 3.2, we have that the vector $f = \phi$ is a multiple of the unique eigenvector z of the primitive matrix $\mathbf{A}$ to the eigenvalue $\lambda = 1$. Therefore, the equations (8) reduce in steady state to

$$0 = z_k\beta_k - \sum_j A_{kj}z_j\beta_j - (\mathfrak{c}_k + \mathfrak{p}_k)z_k + \delta_{k0}\sum_j(\mathfrak{c}_j + \mathfrak{p}_j)z_j,\ k = 0:K \quad (9)$$

We define the vectors y and b by

$$y_k = \beta_k z_k,\ k = 0:K, \quad b = (-\overline{b}, (\mathfrak{c}_1 + \mathfrak{p}_1)z_1, ..., (\mathfrak{c}_K + \mathfrak{p}_K)z_K),$$

$$\overline{b} = \sum_{j=1}^{K}(\mathfrak{c}_j + \mathfrak{p}_j)z_j\ .$$

With this definition, equation (9) becomes

$$(I - \mathbf{A})y = b\ .$$

Since the vector b is orthogonal to the left eigenvector $(1, .., 1)^T$ of the primitive Markov matrix $\mathbf{A}$, and therefore to the nullspace of $I - \mathbf{A}$, there is a solution. This solution is given by $y = y^p + \rho z$, i.e. by a particular solution y^p and an arbitrary multiple of the eigenvector z. This implies that, given a set of $\mathfrak{c}_k$'s and $\mathfrak{p}_k$'s, any set of prices $\beta_k, k = 0:K$ satisfying (9) is given by

$$\beta_k = \frac{y_k^P}{z_k} + \rho\ k = 0:K,\ ,$$

for an arbitrary value of ρ. ρ has to be chosen large enough, such that the unit prices β_k are positive.

∎

So, in summary, prescribing a certain distribution matrix $\mathbf{A}$ uniquely defines - up to a multiplicative constant - the steady state flows in the system. Given the flow, we can always create an order policy by, choosing a matrix $\mathbf{\Omega}$, that produces this flow. Given the flow and the orders, we can always create a price structure, by choosing the vector β which yields an arbitrary amount of profit - at the expense of the final customer.

4 A mixed integer programming approach

In this section we take, in some sense, the opposite approach as in Section 3. Given the profit as a cost functional, we choose the distribution matrix $\mathbf{A}$ and the order matrix $\mathbf{\Omega}$ dynamically optimize the functional. The solution to this problem can of course only be given numerically. We start, by discretizing the dynamical system defined in Section 2. On the discrete time level, the dynamics will actually appear as a large set of nonlinear constraints for a given cost functional. There are essentially two ways to treat the the resulting constrained optimization problem: Either the constraints are formulated on a continuous level using an adjoint calculus to compute a restricted gradient direction as in [11] ,[13], or a nonlinear programming approach is used, introducing additional binary variables. The latter leads to a mixed integer program MIP [6], [9].

4.1 Discretization

We start with a proper discretization of the differential equations (2) and (3) for the inventories p_k and q_k. We discretize on a uniform mesh in time setting $p_k^n = p_k(n\Delta t),\ n = 0 : N$ with $T = N\Delta t$ the final time of the simulation. As is almost always the case in relaxation models, we have introduced an artificial $O(\frac{1}{\varepsilon})$ time scale in the system, and thus artificially created a stiff system. In order not to impose too severe a restriction on the time step Δt we discretize equation (2) implicitly, giving

$$p_k^n = p_k^{n-1} + \Delta t \max\{\psi_k^{n-1} - \mu_k, -\frac{p_k^n}{\varepsilon}\},\ n = 1 : N\ , \tag{10}$$

with ψ_k^{n-1} the influx at the previous time step. Equation (10) can be inverted explicitly for p_k^n, by using the following

Lemma 4.1 *The function $u(x) = \min\{ax + b, cx + d\}$ is invertible for $a > 0,\ c > 0$. This inverse is given by $u^{-1}(y) = \max\{\frac{y-b}{a}, \frac{y-d}{c}\}$.*

Proof: The function u is strictly monotonically increasing, and therefore has a functional inverse u^{-1}. Let w.l.o.g. $a > c > 0$ hold. Then we have

$$u(x) = cx + d\ for\ x \le x_0 = \frac{d-b}{a-c}, \quad u(x) = ax + b\ for\ x \ge x_0,$$

The inverse $u^{-1}(y)$ is therefore given by

$$u^{-1}(y) = \begin{pmatrix} \frac{y-d}{c} & \text{for} & y \le y_0 = u(x_0) = \frac{ad-cb}{a-c} \\ \frac{y-b}{a} & \text{for} & y \ge y_0 \end{pmatrix}$$

or, equivalently

$$u^{-1}(y) = \begin{pmatrix} \frac{y-d}{c} & \text{for} & \frac{y-d}{c} \geq \frac{y-b}{a} \\ \frac{y-b}{a} & \text{for} & \frac{y-b}{a} \geq \frac{y-d}{c} \end{pmatrix} .$$

Therefore, we have

$$u^{-1}(y) = \max\{\frac{y-b}{a}, \frac{y-d}{c}\} .$$

■

Reordering equation (10), we have

$$\min\{p_k^n - \Delta t(\psi_k^{n-1} - \mu_k), (1 + \frac{\Delta t}{\varepsilon})p_k^n\} = p_k^{n-1}$$

and therefore, setting $a = 1,\ b = -\Delta t(\psi_k^{n-1} - \mu_k),\ c = 1 + \frac{\Delta t}{\varepsilon},\ d = 0$ in Lemma 4.1, we obtain

$$p_k^n = \max\{p_k^{n-1} + \Delta t(\psi_k^{n-1} - \mu_k), \frac{p_k^{n-1}}{1 + \frac{\Delta t}{\varepsilon}}\} = p_k^{n-1} + \Delta t \max\{\psi_k^{n-1} - \mu_k, -\frac{1}{\varepsilon + \Delta t} p_k^{n-1}\}$$

which we write as

$$p_k^n = p_k^{n-1} + \Delta t(\psi_k^{n-1} - \phi_k^{n-1})$$

with

$$\phi_k^{n-1} = \min\{\mu_k, \psi_k^{n-1} + \frac{1}{\varepsilon + \Delta t} p_k^{n-1}\} \tag{11}$$

Therefore the implicit discretization of equation (2) can be written in explicit form as

$$p_k^n = p_k^{n-1} + \Delta t(\psi_k^{n-1} - \phi_k^{n-1})$$

with the numerical outflux ϕ_k^{n-1} given by (11). Note, that this eliminates any restriction on the time step, since the flux function ϕ_k^{n-1}, as defined in (11), is well defined in the limit $\varepsilon \to 0$. We employ the same implicit discretization strategy for the evolution of the output inventory q_k in (3). To avoid any additional interpolation procedure, we assume that all the processing times $\tau_k,\ k = 0 : K$ are integer multiples of the time step Δt. So $\frac{\tau_k}{\Delta t} \in \mathbb{N}$ holds. Thus, we obtain, setting

$$\psi_k^n \to \phi_k^{n-\tau_k/\Delta t}, \quad \mu_k \to \omega_k^{n-1} ,$$

$$(a)\ q_k^n = q_k^{n-1} + \Delta t(\phi_k^{n-\tau_k/\Delta t} - f_k^{n-1}), \tag{12}$$

$$(b)\ f_k^{n-1} = \min\{\omega_k^{n-1}, \phi_k^{n-\tau_k/\Delta t} + \frac{1}{\varepsilon + \Delta t} q_k^{n-1}\}$$

Any optimization problem, involving the discretization of the dynamical system, as formulated above, will still be nonlinear, because of the nonlinear (or, more precisely, piecewise linear) definition of the flux functions in (11) and (12). The goal of this section is to formulate an optimization problem for the dynamics defined in Section 2 in a framework close to a linear programming problem, i.e. as a mixed integer programming problem. A mixed integer program (MIP [6]) is a linear program which includes binary switches, i.e. variables taking only values in $\{0, 1\}$. The advantage of linear and mixed integer programming approaches is, that they are capable of dealing with an enormous amount of free variables. The basic tool to convert an optimization problem involving the piecewise linear flux functions (11) - (12) into a MIP is given by the following Lemma (see [9]):

Lemma 4.2 *Let $\xi \in \{0, 1\}$ be a binary variable. Let $M > |a - b|$ be a sufficiently large constant. Then, for a given constant M, the solution of the inequalities*

$$a - M\xi \leq \phi \leq a, \quad b - M(1 - \xi) \leq \phi \leq b \tag{13}$$

is given by

$$\phi = \min\{a, b\}\ .$$

Proof: Since ξ is a binary variable, there are two ways to satisfy the inequalities (13), namely

- For $\xi = 0$: $a = \phi,\ b - M \leq \phi = a \leq b$.
- For $\xi = 1$: $b = \phi,\ a - M \leq \phi = b \leq a$.

In both cases, we have $\phi = \min\{a, b\}$.

■

Using Lemma 4.2, we replace the definition (11)-(12) of the fluxes ϕ_k^n, f_k^n by the constraints

$$(a)\ \mu_k - M\xi_k^n \leq \phi_k^n \leq \mu_k, \tag{14}$$

$$(b)\ \psi_k^n + \frac{p_k^n}{\varepsilon + \Delta t} - M(1 - \xi_k^n) \leq \phi_k^n \leq \psi_k^n + \frac{p_k^n}{\varepsilon + \Delta t}$$

$$(c)\ \omega_k^n - M\eta_k^n \leq f_k^n \leq \omega_k^n,$$

$$(d)\ \phi_k^{n-\tau_k/\Delta t} + \frac{q_k^n}{\varepsilon + \Delta t} - M(1-\eta_k^n) \le f_k^n \le \phi_k^{n-\tau_k/\Delta t} + \frac{q_k^n}{\varepsilon + \Delta t}$$

with the binary variables $\xi_k^n, \eta_k^n \in \{0,1\},\ k = 0:K,\ n = 0:N$, and a constant M, chosen a priori sufficiently large.

The MIP approach allows us to optimize the order strategies, given by the matrix $\mathbf{\Omega}$, as well as the distribution strategies, given by the matrix $\mathbf{A}$ dynamically at the same time. In order to encode the topology of the network, we define the elements of the order matrix $\mathbf{\Omega}$ as $\Omega_{jk}^n = \theta_{jk}\tilde{\Omega}_{jk}^n$, where the matrix $\Theta = \{\theta_{jk},\ j,k = 0:K\}$ denotes the adjacency matrix of the graph defining the network topology, i.e. $\theta_{jk} = 1$ if node number j can order from node number k, and $\theta_{jk} = 0$ otherwise. Similarly, we define $F_{jk}^n = \theta_{jk}\tilde{F}_{jk}^n$ for the fluxes. In the context of the MIP approach, the orders and fluxes $\tilde{\Omega}_{jk}^n, \tilde{F}_{jk}^n$ at each time step are treated as free variables to be optimized. In order to guarantee conservation of product, we have to add the constraint

$$f_k^n = \sum_{j=0}^{K} \theta_{jk}\tilde{F}_{jk}^n, \quad \psi_k^n = \sum_{j=0}^{K} \theta_{kj}\tilde{F}_{kj}^n, \quad k = 0:K,\ n = 0:N\ , \tag{15}$$

and in order to guarantee that fluxes cannot exceed orders, we enforce the constraints

$$0 \le \tilde{F}_{jk}^n \le \tilde{\Omega}_{jk}^n, \quad k = 0:K,\ n = 0:N\ . \tag{16}$$

Of course, only the orders and fluxes for adjacent nodes, i.e. for nodes j and k for which $\theta_{jk} = 1$ holds, have to be used in the actual program. The solution of the MIP implicitly defines an, adaptive and time dependent, distribution policy matrix $\mathbf{A}$, given by

$$A_{jk}^n = \frac{\theta_{jk}\tilde{F}_{jk}^n}{f_k^n}\ ,$$

and the amount of orders received by node number k, which is used in the constraint (14)(b), is given by

$$\omega_k^n = \sum_j \theta_{kj}\tilde{\Omega}_{kj}^n\ . \tag{17}$$

So, altogether, the external variables, which have to be supplied to the MIP, are

- μ_k the processor capacities.

- θ_{jk} the adjacency matrix of the graph.
- $\tilde{\Omega}^n_{Kj}$ the time dependent orders of the final customer.
- p^0_k, q^0_k, the initial inventory positions.
- $\phi^n_k,\ n = -\tau_k/\Delta t : 0$, the past influx of the processors, defining the processor contents at time $t = 0$.

The free variables, to be optimized consist of

- $\tilde{\Omega}^n_{jk},\ j = 1 : K-1,\ k = 1 : K$: the internal orders
- $\tilde{F}^n_{jk}, f^n_k, \psi^n_k, \phi^n_k, \omega^n_k$: the partial and total fluxes and total orders received for each node, given in terms of the $\tilde{\Omega}^n_{kj}$ by the constraints (14)-(17)
- $\xi^n_k,\ \eta^n_k$: the auxiliary binary variables, used in the MIP formulation.

There are various possible goals to be followed when defining the cost functional to be optimized. The simplest one, which is the only one considered in this paper, is to optimize the capitalization of all the interior nodes at the final time. So, the cost functional is of the form

$$\mathcal{J} = \sum_{k=1}^{K} \kappa^N_k \tag{18}$$

where the capital κ^n_k of node k at time $t = n\Delta t$ satisfies, according to (7),

$$\kappa^n_k = \kappa^{n-1}_k + \Delta t(\beta_k f^{n-1}_k - \sum_{j=0}^{K} A_{kj}\beta_j f^{n-1}_j - \alpha_k \phi^{n-1}_k), \quad n = 1 : N\ . \tag{19}$$

In order to model bankruptcies, we make the possible orders dependent of the capitalization rates of the individual nodes, and force each node to cease ordering as soon as its capital falls below a certain threshold $\underline{\kappa}_k$. In the context of the MIP approach, we implement this by introducing another binary variable $\nu^n_k \in \{0, 1\}$ and by defining the total rate at which node k orders as

$$\sigma^n_k = \sum_{j=0}^{K} \theta_{kj}\tilde{\Omega}^n_{kj}\ . \tag{20}$$

To force the node into bankruptcy as soon as its capital falls below the threshold, we add the constraints

$$\sigma_k^n \leq M\nu_k^n, \quad M(\nu_k^n - 1) \leq \kappa_k^n - \underline{\kappa}_k \leq M\nu_k^n \ , \tag{21}$$

where M again denotes a sufficiently large a priori constant. Again, as in the Lemma 4.2, there are two ways to satisfy the constraint (21). For $\nu_k^n = 1$ we have $0 \leq \kappa_k^n - \underline{\kappa}_k$ and essentially no constraint for σ_k^n (provided that M is sufficiently large). For $\nu_k^n = 0$, the bankruptcy case, we have $\kappa_k^n - \underline{\kappa}_k \leq 0$ and $\sigma_k^n = 0$. Adding the variables κ_k^n and σ_k^n to the the system together with the constraints (19)-(21), allows now for the optimization of the cost functional (18) together with the possibilities of bankruptcies.

Combining all discretization we observe that the optimization problem is in a fact mixed–integer programming problem and given by

$$\max \ (18) \text{ subject to } (10) - (21). \tag{22}$$

5 Numerical results

For a numerical study of the optimal controls we consider three different scenarios $\mathbf{I} - \mathbf{III}$.

I As 'benchmark' scenario we denote the problem as stated in equation (22). This solution yields the maximal possible profit since there are no additional constraints on the distribution and order rates.

II We require the order policies to be time–independent. This amounts to add the constraints

$$\Omega_{jk}^{n+1} = \Omega_{jk}^n. \tag{23}$$

This choice is reasonable if the suppliers do not want to change there policy dynamical. Clearly, this additional constraint restricts the set of possible solutions. The optimization problem hence reads

$$\max \ (18) \text{ subject to } (10) - (21) \text{ and } (23) \tag{24}$$

and this scenario will be called 'time–independent orders' in the numerical results.

III We impose the following rule: Whenever the supplier S_k is *not* bankrupt, the supplier has to order up to his capacity μ_k:

$$\sum_j \Omega_{jk}^n \leq \nu_k^n \mu_k. \tag{25}$$

This rule is motivated by the fact that the complete production line should have the highest possible utilization. Therefore, whenever a supplier has a positive cash flow, he should order as much goods as possible. The optimization problem hence reads

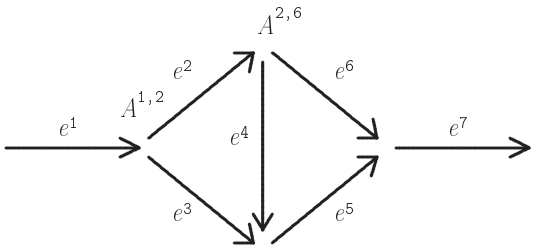
$$\max \ (18) \text{ subject to } (10) - (21) \text{ and } (25) \tag{26}$$

and this scenario will be called 'order–up to capacity' for short.

The optimization problem is solved using a mixed–integer formulation of the previous model. The mixed–integer problem is solved using the commercial software ILOG CPLEX V11.0 [12] with default parameters. We study the behavior of the optimal controls Ω_{jk} and A_{jk} on two different networks and the three different cases $\mathbf{I} - \mathbf{III}$. We use default parameters when running the commercial solver and require a maximum computation time of $24h$. All computations are done on a AMD 2 Ghz personal computer. We further set $\tau_k \equiv \Delta t \equiv \epsilon \equiv 1$ for all examples. The further parameters are given in the subsequent sections.

5.1 Computational results for a diamond network

We consider a network with seven suppliers and six vertices. We have a single customer and a single raw material supplier. The possible distribution rates are shown in figure 5.1.



Each supplier (except for customer $k = 7$ and raw material supplier $k = 1$) has specific prices, production capacities and production costs as given in table 1. For each scenario we consider a constant inflow $\psi_1^n = 2$ and a threshold for going bankrupt of $\underline{\kappa}_k = -5$. The total simulation time is $T = 40$. Note that, for the data given the processing chain 'raw material supplier $\rightarrow$ supplier 2 $\rightarrow$ supplier 6 $\rightarrow$ customer' is the preferred one. However, the inflow of two parts per time cannot be passed through supplier 6 and there is the possibility to either store the goods in the input inventory of 6 or redistribute along suppliers 4 and 5.

Processor k	β_k	μ_k	α_k
2	1	2	0
3	2	1	1
4	1	1	1
5	1	1	10
6	1	1	10

Table 1: Specification of suppliers S_k present in the diamond network.

We give computational results for the scenarios **I-III** by solving (22), (24) and (26), respectively. In table 2 we report on the size of the optimization problem (non–zero variables), the computational time used by CPLEX (CPU time), the optimal profit $\sum_k \kappa_k(T)$ and the amount of delivered parts over time at customer, i.e., $\int_0^T f_7(t)dt$. In table 3 we report on percentage of the time the supplier has been bankrupt during the simulation. Finally, we present the dynamic behavior of the total money flow and the delivered parts in figure 1 for all three cases. The total number of delivered parts as reported in table 2 is the integral of the corresponding functions over time.

We offer the following interpretation of the results: Clearly, the benchmark solution yields the maximal profit, but the number of delivered parts is the highest in case **III**, since the policy requires to order as many parts as possible. In the benchmark case some internal suppliers go bankrupt for a very long time. However, the total profit for this choice of orders and distribution is still higher than for all other cases. The optimal choice in the benchmark case is to accept the bankruptcy of most of suppliers in order to maximize the total profit. In the case of the time–dependent policy most of the suppliers to do not go bankrupt, however, there is nearly no part delivered and the overall profit is the least of all cases. In figure 1 we observe that it takes some time for the parts to be processed in the supply chain and there are frequently regions where no part is delivered. This regions correspond to bankrupt suppliers who cannot order any further parts and the production stops.

5.2 Computational results for cascade network as in [5]

We consider a network as introduced by Battiston et. al. in [5], Section 2.1. Therein, they introduced a network of connected suppliers, depicted schematically in Figure 2, and studied bankruptcy and production failures. We adopt their geometry, adding a raw material supplier and a

	Benchmark (**I**)	Time–independent orders (**II**)	Order up to capacity (**III**)
# nonzero vars	1497	1907	1600
CPU–times [sec]	3480	406	140
Optimal profit	260,4	16,72	31,00
Delivered parts	3,6	1,54	4,00

Table 2: Comparison of computation results for the diamond network.

Supplier	Benchmark (**I**)	Time–independent orders (**II**)	Order up to capacity (**III**)
2	72,5%	0 %	57,5 %
3	37,5 %	0 %	0 %
4	0 %	0 %	0 %
5	5 %	2,5 %	7,5 %
6	5 %	2,5 %	7,5 %

Table 3: Percentage of time a supplier has been bankrupt.

customer to simplify the presentation of the results. We have a total of 74 suppliers at three stages. Each supplier is connected to three other suppliers as in [5]. We note that this rather specific topology, where nodes can only order from 'neighboring' nodes at the previous production level, corresponds essentially to the rudimentary discretization of a diffusion equation. The parameters of the supplier are as follows: All suppliers S_k have a processing capacity of $\mu_k = 1$, production and prices of $\beta_k = \alpha_k = 1$ except for those five suppliers represented by a dashed line in figure 2. Those five suppliers have production costs $\beta_k = 5$, capacities $\mu_k = 1$ and prices $\alpha_k = 1$. The raw

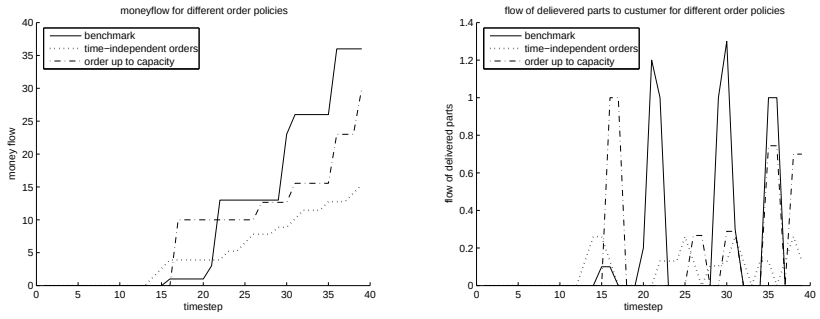


Figure 1: Profit $t \to \sum_k \kappa_k(t)$ over time and delivered parts to the customer over time $t \to f_7(t)$

material supply is $\psi_1^n = 9$ and $T = 40$. We show a non–optimal and an optimal situation in figure 3 for the benchmark problem (22). Depending on the cumulative load of the supplier S_k, i.e., $\int_0^T f_k(t)dt$, the arcs are displayed in bold or light type. Unused suppliers are not shown. We observe that the optimal distribution prevents using the expensive suppliers at the bottom right of the network (indicated by a red dashed line in figure 2). The discretized optimization consists of 17908 real variables and 8880 binary variables. In the non–optimal case we have 11807 non–zero variables in the solution and the feasible solution is determined in 13,94 seconds. The optimal solution has 6047 non–zero variables and solution time has been 50,89 seconds.

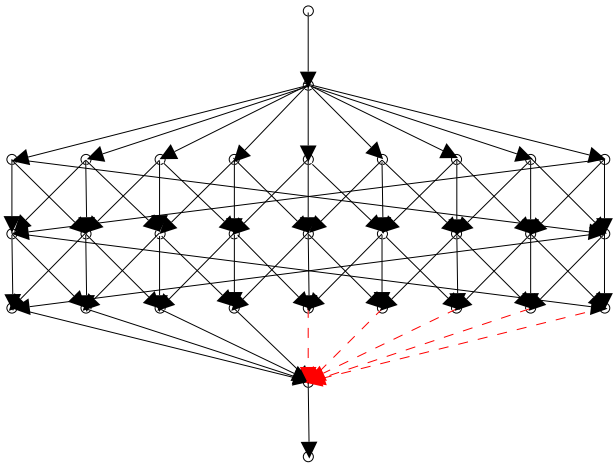


Figure 2: A cascade of three levels with a total of 74 suppliers. Every supplier is connected to three further suppliers. The suppliers represented by a dashed red line have different production costs.

6 Summary

In this paper we presented a model for a production network including order and distribution policies and money flow. The model is an extension to recently proposed continuous production network models. The policies are determined by an optimization problem for maximizing the money flow where the discretized maximization problem is solved by mixed–integer programming. We compared the optimal policies to other a priori given policies.

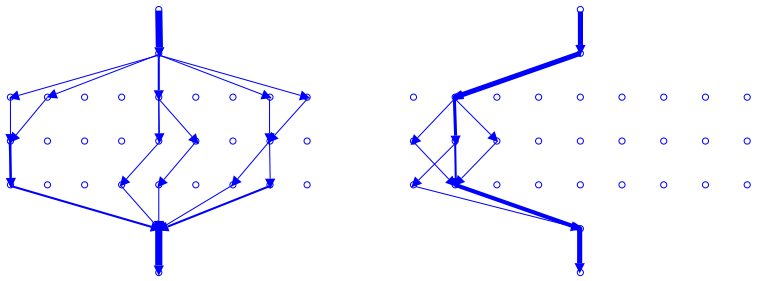


Figure 3: Non–optimal (left) and optimal load (right) of each supplier over time. Load is represented as cumulative flow $\int_0^T f_k(t)dt$. A bold type arrow corresponds to high production load.

Acknowledgements

This work was supported by NSF grant DMS-0604986, the DAAD project D/06/28176 and the DFG grant HE 5386/6-1. Special thanks go to Sebastian Kühn for his help in generating sample scenarios and implementing various mixed–integer problems.

References

[1] D. Armbruster, C. de Beer, M. Freitag, T. Jagalski, C. Ringhofer, *Autonomous Control of Production Networks using a Pheromone Approach*, PHYSICA A, 363(1):104-114, 2006.

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

[3] D. Armbruster, D. Marthaler, C. Ringhofer, *Kinetic and Fluid Model Hierarchies for Supply Chains*, SIAM J. on Mutiscale Modeling and Simulation, 2:43-61, 2004.

[4] Bak P., Chen K., Scheinkman J. and M. Woodford, *Aggregate Fluctuations from Independent Sectoral Shocks: Self- Organized Criticality in a Model of Production and Inventory Dynamics*, Ricerche Economiche, 47:3-30, 1993.

[5] Battiston, S., Delli Gatti, D., Gallegati, M., Greenwald, B., Stiglitz, J.E., *Credit chains and bankruptcy propagation in production networks,* Journal of Economic Dynamics and Control, 31(6):2061-2084, 2007.

[6] R.E. Bixby, M. Fenelon, Z. Gu, E. Rothberg and R. Wunderling, *MIP: Theory and Practice – Closing the Gap*, System Modelling and Optimization, 19 – 50, 1999.

[7] Caldarelli G., Battiston S., Garlaschelli D., Catanzaro M., *Emergence of Complexity in Financial Networks,* Lecture Notes in Physics, Volume 650:399 - 423, "Complex Networks", editors: Eli Ben-Naim, Hans Frauenfelder, Zoltan Toroczkai, Springer-Verlag, 2004.

[8] A. Fügenschuh, M. Herty, A. Klar and A. Martin, *Combinatorial and Continuous Models for the Optimization of Traffic Flows on Networks*, SIAM J. Optimization 16(4):1155 - 1176, 2006.

[9] A. Fügenschuh, S. Göttlich, M. Herty, A. Klar and A. Martin, *A mixed–integer programming approach for the optimization of continuous models in supply chain management*, preprint 2006.

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

[11] M. Herty, C. Ringhofer, *Optimization For Supply Chain Models With Policies* PHYSICA A, 380: 651-664, 2007.

[12] ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451, USA, *Information available at URL http://www.cplex.com.*

[13] C. Kelley, *Iterative methods for optimization*, SIAM Frontiers in Applied Mathematics, 1999.

[14] G. Latouche, V. Ramaswami, *Introduction to Matrix Analytic Methods,* in *Stochastic Modelling*, 1st edition. Chapter 2: PH Distributions; ASA SIAM, 1999.

[15] G. Nemhauser and L.A. Wolsey, *Integer and Combinatorial Optimization,* Wiley-Interscience, 1999.