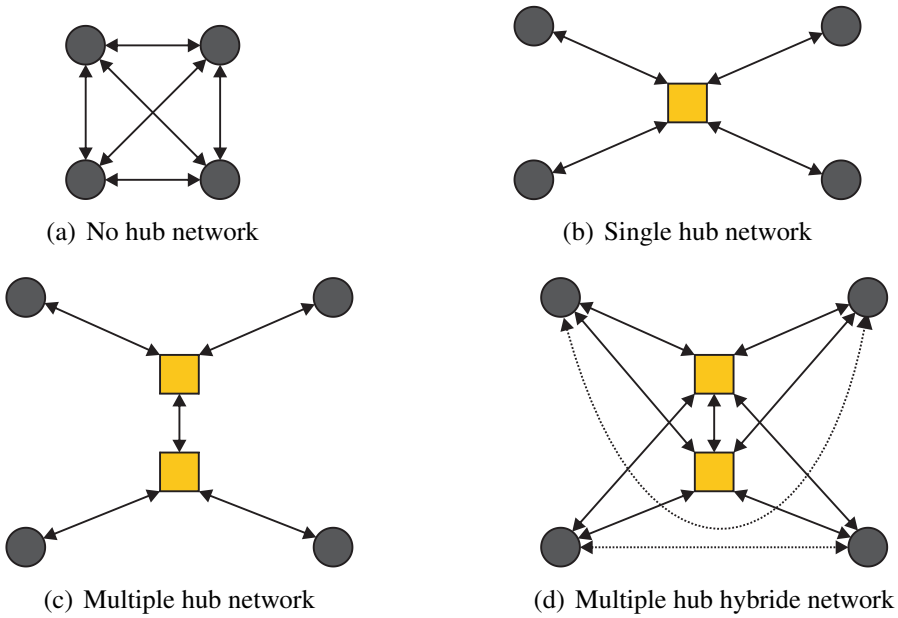
Contents

1 Introduction

The optimal design and/or the optimal control of dynamic stochastic systems are an actual and interesting problem for theoreticans as well as practitioners. At present logistics is one of the most important application areas. There exist very different formulations and models to optimize logistic systems. In our report we will concentrate on some aspects of how to use transportation resources in an optimal way, i. e., we will discuss solution approaches to the problem of the optimal size and allocation of transportation resources. In the logistic literature we can find manifold models and solution methods dealing with that problem. To optimize a logistic system leads as a rule to very complex decision problems. Therefore corresponding models suffer from various simplification assumptions. For instance, most models assume deterministic known demand for transportation services. In this case linear and non-linear network programming models are applied (cp. [WHW99]). More sophisticated models and solution methods assume stochastic demand. Here models from inventory and queueing theory are appropriate ([DH97], [KL76]). However, in all cases no closed-form solutions are available. The majority of papers is dealing with algorithms for approximate solutions in the discrete-time case with known demand and infinite transportation capacity. For more realistic models and solutions the broadest applicable approach is simulation. However, simulation is not an optimization tool. But we can combine simulation and optimization and apply the simulation optimization approach. [Fu94] gives a good overview on simulation optimization. The first paper, which applies that approach to some logistic problems seems to be [KKN03]. In the present report we are interested in solutions for the so-called fleet-sizing-and-allocation problem (FSAP). The FSAP is one of the most interesting and hard to solve logistic problems. It consists of two interdependent parts - to define the optimal size of a transport fleet and to reallocate empty transportation units among the locations of a logistic network. A transportation unit (TU) in our sense is a reusable unit for the realization of a given kind of transportation service. Such units may be cars, trucks, rail cars or air planes as well as containers and material handling equipment. To make the FSAP a little bit more tractable we concentrate on logistic systems with a special hub-and-spoke structure.

Generally, in a hub-and-spoke transportation network a centralized planner has to find freight routes, frequency of service, type of TU's to be used, and transportation volumes. Such a structure can be profitable only if there exists concentration of freight volume on some service links in the network. Therefore hub-and-spoke networks are very actual and can be found e.g. in air freight distribution, two-echelon inventory systems, container distribution systems and others. Such networks have been developed with intention to improve competitiveness, efficiency and effectiveness of their operations. The main motivation for such network structure lies in savings in transportation cost due to concentration of volume at the hubs.

Hub-and-Spoke network can be categorized into four types depending on whether hub(s) should be visited between source and destination as illustrated in Figure 1. TU's movement originated from a source can reach a destination without visiting any hub (Figure 1(a)); go to a hub from which it moves to its destination (Figure 1(b)); go to a hub from which it moves to another hub after being consolidated with other TU's and then moves to its final destination (Figure 1(c)); some TU's go to their destination directly whereas the others go through one or two hubs (Figure 1(d)).



(a) No hub network (b) Single hub network

(c) Multiple hub network (d) Multiple hub hybride network

Figure 1: Hub-and-spoke networks

Hubs are facilities that serve as switching in transportation and telecommunication networks. Hubs may be large facilities such as airports or postal sorting centers, or small switching devices in telecommunications networks. At present exist different research directions for hub-and-spoke networks. Many papers are dealing with the hub location problem (e.g. [Cam96], [OSKSK95], [OB98]). It seems highly unlikely that such problem would be solved efficiently as the problem size increase since the problem is known to be NP-hard (see [MW84]). Other authors concentrate on different application areas as air transportation networks ([Ayk95], [JSY83]), railroad networks ([Ass80], [CR86], [Kea92], [NY00]), or flexible manufacturing systems ([**?**]).

In the present report we concentrate on the fleet-sizing-and-allocation problem for single hub networks. The network consists of $M+1$ locations. Location 0 represents a single central, the hub, whereas the locations 1 to M are the spokes. The spokes generate a deterministic or random demand for transportation orders. To serve such an order one of a given number K of TU's, located in the hub, can be used. After realising a transshipment to a spoke the transportation unit returns to location 0. The basic problem is to find an optimal in a given sense number K of TU's and their allocation to transportation orders.

Our presentation is organised as follows. In Section 2 we define a rather general decision problem. Next in Section 3 we investigate a simple model with deterministic demand, fixed number and fixed capacity of TU's. We consider two cases - renting of additional TU's from outside the system is not allowed respectively is allowed. For that simple model we can prove some interesting results on the optimal allocation. In Section 4 we apply queueing theory methods to solve the allocation problem in case of stochastic demand in the spokes. But all these models include some restrictive assumptions, which in general are not fulfilled in reality. Since the FSAP is hardly analytically tractable without these simplifying assumptions we refer to simulation optimization for more realistic models. This topic is considered in Section 5, where we briefly discuss the principle of simulation optimization. We describe also a simulator for hub-and-spoke systems and its combination with an optimisation tool. Some examples show the applicability of our simulation optimization approach.

2 Description of a basic model and decision problem

In the following we imagine the hub and spoke system (HAS system) as a supply network for consumable products or services. Fundamental modelling assumptions are related to the

time flow , which can be continuous or discrete, whereby the elements of the model can be stationary or non-stationary;

planning horizon - finite or infinite, rolling or fixed;

available information - full, partial (adaptive model), no information;

decision/control possibilities - ordering policies, policies for transportations between the warehouse and the retailers and between the retailers, number of TU's, leasing of additional TU's from outside or letting of own TU's, handling of transportation orders (priorities), and satisfaction of the demand;

goal function - long-run average cost, the total discounted cost, average waiting times, fill rates or service rates, and so on. Thereby the goal function may include multiple criterions or a single criterion.

With respect to the elements of a model we assume the following:

1. There exists a central warehouse, the hub, and M retailers, the spokes. The central warehouse stores the products, which can be ordered by the retailers, which have to serve the demand for these products. The warehouse else can order product from producers or produce the products itself.
2. The central warehouse is described by

- number of products,
- storage capacities (for all products or for each single product),
- ordering policy,
- number of own TU's,
- allocation policy of own TU's and leased TU's,
- cost and gain structure.

3. For the retailers we have to define
   - the demand process, which can be stochastic or deterministic,
   - the acceptance-rejection rule for arriving demand,
   - the ordering policy,
   - the storage capacities,
   - the pooling scheme for lateral transports between the retailers,
   - the cost and gain structure.

   The retailers may be identical or different with respect to these characteristics.
4. The TU's we can divide into classes. Such a class is characterized by
   - transportation times (deterministic or random, identical or different),
   - transportation capacities (measured in product units),
   - transportation cost (fixed cost, volume and time proportional cost, other functions),
   - standstill cost for depreciations and so on.

   Each TU moves from the warehouse to a retailer and back. On the return way a TU can transport for instance packing material and/or empty containers.
5. Lead times can exist for the supply of products to the warehouse, for transportation of product to and between retailers. These lead times may be zero, deterministic or random.
6. The cost and gain structure includes
   - ordering cost,
   - fixed cost per time unit and TU for own TU's,
   - rental cost per time unit and rented TU,
   - cost for a transportation per time unit and TU,
   - waiting cost/shortage cost per time unit for transportation orders and demand,
   - holding cost for stored product in the warehouse and the retailers,
   - profit from sold product.

We finish the description of model elements with some remarks on the pooling scheme and the acceptance-rejection rule (AR-rule). The AR-rule handles the arriving demand at retailers. In the backordering case all demand is accepted, whereas in the lost-sales case arriving demand, which cannot be satisfied by available inventory at a retailer, is rejected and lost. In the intermediate case exists a waiting queue with finite capacity for waiting demand. To increase the quality of service for the whole system the retailers and their inventories on hand may be pooled. In case of shortage of product at one retailer and available product at another one lateral transportations between retailers of the same pool are an alternative. The pooling scheme should describe as well the pooling of retailers into groups as the pooling of the inventories at the retailers. The realisation of lateral transportations requires additional resources. If the lead times for transports from the warehouse to retailers are high compared with those between retailers then pooling may be decrease cost and increase the quality of service. However, pooling complicates the problem to control the whole system.

Now let us formulate a general decision respectively optimisation problem as the problem to define for a HAS system

a) the number of own TU's,

b) the number of rented TU's,

c) the ordering of the central warehouse,

d) the release of transportation orders by the retailers, and

e) the allocation of TU's to transportation orders

in such a way that given performance criterions will be optimized.

From the above description of model elements and the formulated general optimization problem it is obvious, that in dependence of the concrete assumptions to the elements of a model we get a great variety of different models and problems. Most of them are not analytically tractable. In the following sections we will consider some models, which we have investigated and for which we have at least some algorithms for getting a solution. We will move from the simplest case to more realistic models, i.e., from a single-period model with deterministic demand to a queueing model with continuous time.

3 A single-period, deterministic-demand model

We consider the HAS system with a single-period planning horizon, deterministic demand, full information, and total cost criterion. To be concrete we assume the following:

1. There are a single warehouse and M retailers.
2. The warehouse has an ample amount of a single product and K TU's.
3. The demand at retailer i during the period is known and equal to $d_i, i = 1,\ldots,M$. Each retailer has zero inventory and can order only full TU's. The cost structure comprises for retailer i
   p_i - shortage cost per unit not satisfied demand,
   h_i - holding cost per unit not sold product.
4. The K TU's belong to a single class, which is characterized by negligible transportation times, transportation capacity Q, and transportation cost c_i per TU going to retailer i.
5. We have no lead times.

The problem is to allocate the K TU's to the retailers such that the total cost will be minimized.

3.1 Solutions for the allocation problem without renting possibilities

To formalize the corresponding optimization problem we define

$n_i \in \mathbb{N} = (0,1,2,\ldots)$ the number of TU's allocated to retailer i,
$\mathbf{n} = (n_1, n_2, \ldots, n_M) \in \mathbb{N}^M$ the allocation-vector,
$k(\mathbf{n})$ the total cost for the period under allocation-vector$\mathbf{n}$.

It holds that

$$k_i(n_i) = c_i \cdot n_i + \begin{cases} p_i \cdot (d_i - n_i \cdot Q), & n_i \cdot Q \leq d_i; \\ h_i \cdot (n_i \cdot Q - d_i), & n_i \cdot Q \geq d_i, \end{cases} \tag{3.1}$$

and

$$k(\mathbf{n}) \;=\; \sum_{i=1}^{M} k_i(n_i) \;\;=\;\; \sum_{i=1}^{M} \left\{ c_i n_i + p_i \left(d_i - n_i \cdot Q\right)^+ + h_i \left(n_i Q - d_i\right)^+ \right\}, \; \mathbf{n} = (n_1, n_2, \ldots, n_M),$$

where $(x)^+ = max(x,0)$ for any real x. With these definitions the optimization problem to be solved can be reformulated as optimization problem (OP-I):

For given $K \in \mathbb{N}$ find an allocation-vector $\mathbf{n}^{(I)}$, which minimizes the total cost, i.e.,

$$\left.\begin{array}{ll} & \text{Minimize } k(\mathbf{n}) \\ \text{s.t.} & \\ & \sum_{i=1}^{M} n_i \leq K \\ & n_i \in \mathbb{N}, i = 1,\ldots,M \end{array}\right\} \text{(OP-I)}.$$

Obviously $k_i(\cdot)$ is an integer-convex function of n_i for $i = 1,\ldots,M$, and, consequently, $k(\cdot)$ is integer-convex in all its arguments. This convexity property considerably simplifies (OP-I). Let $\mathbf{n}^* = (n_1^*, n_2^*, \ldots, n_M^*) \in \mathbb{N}^M$ denote that allocation-vector, which solves (OP-I) for $K = \infty$, i.e., we do not take into consideration that the central warehouse has only K TU's. Since goal function $k(\cdot)$ is additive with respect to its arguments, n_i^* minimizes function $k_i(\cdot)$ for each i. To calculate n_i^* we use the first-order differences $\Delta k_i(\cdot)$, which are defined as

$$\Delta k_i(n_i) := k_i(n_i+1) - k_i(n_i), n_i \geq 0. \tag{3.2}$$

From convexity follows that n_i^* fulfills the two inequalities

$$k_i(n_i^*) \leq k_i(n_i^*+1) \text{ and } k_i(n_i^*) \leq k_i(n_i^*-1)$$

or, equivalently, $\Delta k_i(n_i^*) \geq 0$ and $\Delta k_i(n_i^*-1) \leq 0$. In other words, it holds that

$$n_i^* = argmin\{n : \Delta k_i(n) \geq 0\} = argmax\{n : \Delta k_i(n-1) \leq 0\}. \tag{3.3}$$

The characterization of n_i^* by (3.3) is too general for use. With (3.2) we get from (3.2) that

$$\Delta k_i(n_i) = c_i + \begin{cases} -p_i \cdot Q, & (n_i+1)\cdot Q < d_i \\ h_i \cdot Q - (h_i+p_i)\cdot(d_i - n_i \cdot Q), & n_i \cdot Q \leq d_i < (n_i+1)\cdot Q \\ h_i \cdot Q, & d_i < n_i \cdot Q \end{cases} \tag{3.4}$$

Now we have to distinguish between four cases.

Case 1: $(n_i^*+1)\cdot Q \leq d_i$.
From the optimality conditions for n_i^* and (3.4) follows
$\Delta k_i(n_i^*) = c_i - p_i \cdot Q \geq 0 \geq \Delta k_i(n_i^*-1) = c_i - p_i \cdot Q$, i.e., $\Delta k_i(n_i^*) = \Delta k_i(n_i^*-1) = c_i - p_i \cdot Q = 0$.
This is equivalent to $c_i = p_i \cdot Q$, i.e., the transportation costs for product quantity Q are equal to the shortage costs for the same quantity. In other words, to order product generates the same cost as not to order. Thus we have a trivial solution $n_i^* = 0$. To exclude that triviality we introduce

Assumption (ET) - *Efficiency of Transportation*:

$$c_i < p_i \cdot Q \text{ for } i = 1, \ldots, M.$$

One consequence of assumption (ET) is that in case 1 we get the contradiction

$$c_i - p_i \cdot Q > 0 \geq c_i - p_i \cdot Q,$$

i. e., $n_i^* = 0$ is not optimal.

Case 2: $n_i^* \cdot Q \leq d_i < (n_i^* + 1) \cdot Q$.
Here we get, taking into account assumption (ET), that

$$\Delta k_i(n_i^*) = c_i + h_i \cdot Q - (h_i + p_i) \cdot (d_i - n_i^* \cdot Q) \geq 0 > c_i - p_i \cdot Q = \Delta k_i(n_i^* - 1).$$

Rearranging terms this is equivalent to the optimality condition

$$(n_i^* + 1) - \frac{d_i}{Q} \geq \frac{p_i - \frac{c_i}{Q}}{h_i + p_i} \qquad or \qquad n_i^* = argmin \left\{ n : (n+1) - \frac{d_i}{Q} \geq \frac{p_i - \frac{c_i}{Q}}{h_i + p_i} \right\}.$$

Case 3: $(n_i^* - 1) \cdot Q \leq d_i < n_i^* \cdot Q$.
In the same way as for case 2 we get the optimality condition

$$n_i^* - \frac{d_i}{Q} \leq \frac{p_i - \frac{c_i}{Q}}{h_i + p_i} \qquad or \qquad n_i^* = argmax \left\{ n : n - \frac{d_i}{Q} \leq \frac{p_i - \frac{c_i}{Q}}{h_i + p_i} \right\}.$$

Case 4: $d_i \leq (n_i^* - 1) \cdot Q$.
With assumption (ET) and optimality condition (3.3) we get $0 > c_i - p_i \cdot Q = \Delta k_i(n_i^*) \geq 0$. But this is a contradiction.

We have shown that only the cases 2 and 3 are relevant and that the optimality condition is

$$n_i^* = argmin \left\{ n : (n+1) - \frac{d_i}{Q} \geq \frac{p_i - \frac{c_i}{Q}}{h_i + p_i} \right\} = argmax \left\{ n : n - \frac{d_i}{Q} \leq \frac{p_i - \frac{c_i}{Q}}{h_i + p_i} \right\}. \tag{3.5}$$

Remark 3.1 The optimality condition (3.5) has an important intuitive interpretation. The optimal number of ordered quantities Q is at least equal to the integral part n' of the demand divided by that quantity. If there remains some unsatisfied demand the ordering of an additional TU depends on the amount of unsatisfied demand $d_i - n' \cdot Q$ and the cost factors. Thus the minimization of

function $k_i(\cdot)$ represents the classical inventory problem with the restriction to order sizes equal to multiples of a given quantity Q.

Let us return now to problem (OP-I). It is obvious that $n^{(I)} \equiv n^*$ if $\sum_{i=1}^{M} n_i^* \leq K$. Otherwise we suggest the following procedure to solve (OP-I):

1. Calculate n^* using (3.5).
2. REPEAT
   (i) Chose a retailer i.
   (ii) Decrease n_i^* by one.

   UNTIL $\sum_{i=1}^{M} n_i^* \leq K$.

For the applicability of that procedure we have to answer two questions:

1. How to chose the retailer i in Step 2(i)?
2. Will the search process stop on an optimal solution?

The convexity of the goal function allows applying Marginal Analysis (MA) (see [Fox66]). MA will answer both questions. The given algorithm (MA) and some important properties are given in Appendix A. The formulation there is for the maximization of a concave and strictly increasing function f. We remark that maximization, concavity, and to be strictly increasing for a function f is equivalent to minimization, convexity, and to be strictly decreasing, respectively, of function $(-f)$. Now we put $(-f) = k$ with k from (3.2). Since $k_i(\cdot)$ is an integer-convex function of n_i and since from the definition of n^* follows that $k_i(\cdot)$ is strictly decreasing for $n_i = 0,\ldots,n_i^*, i = 1,\ldots,M$, and algorithm (MA) generates an optimal solution for (OP-I) by Property 3 from the appendix. The general algorithm (MA) substantiates to the

Algorithm *Marginal analysis* for (OP-I) (to solve (OP-I) in case $\sum_{i=1}^{M} (n_i^*) > K$)

INPUT K; $n_i^*, i = 1 \ldots M$.

1. Initial solution $n^{(0)} = (n_1^{(0)},\ldots,n_M^{(0)})$ with $n_i^{(0)} = n_i^*, i = 1,\ldots,M$.
2. $r := 1$.
3. $\mathbf{n}^{(r)} = \mathbf{n}^{(r-1)} - e_i$, where $e_i = (0,0,\ldots,1,0,\ldots,0)$ and i that index, which minimizes $\Delta k_i(n_i^{(r-1)}) := k_i(n_i^{(r-1)} - 1) - k_i(n_i^{(r-1)}),\ n_i^{(r-1)} > 0$.
4. Stop if $r = K - \sum_{i=1}^{M} n_i^*$ Otherwise $r := r + 1$ and go to 3.

OUTPUT Allocation vector $\mathbf{n}$.

In the above given formulation algorithm (MA) is applied "in reverse". From Property 3 it follows that we stop at the optimal solution. Another approach to solve (OP-I) in case $\sum_{i=1}^{M} n_i^* > K$ is dynamic programming, but sometimes with a considerable higher numerical effort.

A stolid application of Marginal Analysis naturally solves (OP-I). But, using the property that $k_i(\cdot)$ is piecewise linear, it is easy to speed up the solution process in case $\sum_{i=1}^{M} n_i^* > K$. For that we sort all retailers such that

$$-\Delta k_1(0) \geq -\Delta k_2(0) \geq \cdots \geq -\Delta k_M(0). \tag{3.6}$$

Since $\Delta k_i(n_i) = c_i - p_i \cdot Q$ for $(n_{i+1})Q \leq d_i$ the inequalities (3.6) are equivalent to

$$p_1 \cdot Q - c_1 \geq p_2 \cdot Q - c_2 \geq \cdots \geq p_M \cdot Q - c_M \tag{3.7}$$

From the inequalities (3.7) follows that to retailer 1 will be allocated TU's until $n_1 \cdot Q \leq d_1$ and $n_1 \leq K$. Thus $n_1' := argmax\{n : n \cdot Q \leq d_1 \text{ and } n \leq K\}$ denotes a first amount of TU's allocated to retailer 1 (cp. Remark 3.1). If $n_1' = K$ then the process stops. Otherwise, in (3.7) we have to replace $p_1 \cdot Q - c_1$ by $-\Delta k_1(n_1') = (d_1 - n_1' \cdot Q) \cdot (h_1 + p_1) - c_1 - h_1 \cdot Q$ and to find for retailer 1 the new place in (3.6) respectively (3.7). If $-\Delta k_1(n_1') \geq p_2 \cdot Q - c_2$ then retailer 1 gets a last additional TU, because of $-\Delta k_1(n_1' + 1) = -(c_1 + h_1 \cdot Q) < p_M \cdot Q - c_M$. Otherwise the process continues with retailer 2 and so on. If in this allocation procedure the retailer to be considered next is an already considered retailer (with a first amount n_i' of allocated TU's) that retailer gets an additional single TU and is no more considered. We describe that allocation procedure by the following algorithm.

Algorithm *Optimal Allocation Procedure (OAP)* (to solve (OP-I) in case $\sum_{i=1}^{M} (n_i^*) > K$).

INPUT K; Q; $d_i, i = 1 \ldots M$.

1. Preliminaries:
   $n_i' = \left\lfloor \frac{d_i}{Q} \right\rfloor, i = 1, \ldots, M$ [1].
2. Initialization:
   $n_i := 0, i = 1, \ldots, M; nsum := 0$.
3. Sorting:
   Define a permutation $i = (i_1, \ldots, i_M)$ of integers 1 to M such that
   $p_{i_1} \cdot Q - c_{i_1} \geq p_{i_2} \cdot Q - c_{i_2} \geq \cdots \geq p_{i_M} \cdot Q - c_{i_M}$.

[1] $\lfloor x \rfloor$ denotes the greatest integer not exceeding x for any real x

4. Iteration:
   WHILE $nsum < K$ DO

   a) Allocation-step:
      IF $n_{i_1} = 0$ AND $n'_{i_1} > 0$
      THEN $n_{i_1} := n'_{i_1}$ and $nsum := nsum + n_{i_1}$
      ELSE $n_{i_1} := n_{i_1} + 1$ and $nsum := nsum + 1$.

   b) Resorting-step:
      Calculate $-\Delta k_{i_1}(n_{i_1})$ and find the new position for retailer i_1 in the permutated sequence of retailers.

   OUTPUT Optimal allocation vector $\mathbf{n}$.

We remark that in case $\sum_{i=1}^{M} n_i^* > K$ algorithm OAP as well as algorithm Marginal Analysis for (OP-I) stops before all retailers are satisfied, i.e. there are some retailers without any delivery, independent of their demand values. Such discrepancies between the demand and the total transporting capacity can lead to very high costs (cp. the examples). To prevent this we next consider a model, where additional TU's can be rented from outside the system.

3.2 Solutions for the allocation problem with renting possibilities

We assume now that the central warehouse can rent additional TU's from outside the system with the same characteristics as the own TU's, but with $c_{r,i}$ as transportation cost for a rented TU going to retailer i. It is natural to assume $c_i \leq c_{r,i}$ for all i. Furthermore, to rent a TU will be profitable only if it leads to decreasing costs. A sufficient condition for that is to complete assumption (ET) by assumption (ER) - *Efficiency of Renting*:

$$c_{r,i} < p_i \cdot Q \text{ for i} = 1, \ldots, \text{M}$$

Let $\mathbf{r} = (r_1, r_2, \ldots, r_M) \in \mathbb{N}^M$ denote the allocation-vector for rented TU's and $k(\mathbf{n}, \mathbf{r})$ the total single-period cost under the two allocation-vectors $\mathbf{n}$ and $\mathbf{r}$. In analogy to (3.1) and (3.2) it holds for $\mathbf{n}, \mathbf{r} \in \mathbb{N}^M$ that

$$k_i(n_i, r_i) = c_i \cdot n_i + c_{r,i} \cdot r_i + \begin{cases} p_i \cdot (d_i - (n_i + r_i) \cdot Q), & (n_i + r_i) \cdot Q \leq d_i; \\ h_i \cdot ((n_i + r_i) \cdot Q - d_i), & (n_i + r_i) \cdot Q \geq d_i, \end{cases} \tag{3.8}$$

and

$$\begin{aligned} k(\mathbf{n},\mathbf{r}) &= \sum_{i=1}^{M} k_i(n_i,r_i) \\ &= \sum_{i=1}^{M} \left\{ c_i n_i + c_{r,i}\cdot r_i + p_i\,(d_i-(n_i+r_i)\,Q)^+ + h_i\,((n_i+r_i)\,Q-d_i)^+ \right\}. \end{aligned} \tag{3.9}$$

Again $k_i(\cdot,\cdot)$ is a piecewise linear, integer-convex function of n_i and r_i for $i=1,\ldots,N$, $k_i(\cdot,\cdot)$ is integer-convex in all its arguments, and we have a second *optimization problem* (OP-II):

For given $K\in\mathbb{N}$ find allocation-vectors $\mathbf{n}^{(II)}$ and $\mathbf{r}^*$, which minimize the total cost, i.e.,

$$\left.\begin{array}{ll} & \text{Minimize } k(\mathbf{n},\mathbf{r}) \\ \text{s.t.} & \\ & \sum\limits_{i=1}^{M} n_i \le K \\ & n_i, r_i \in \mathbb{N}, i=1,\ldots,M \end{array}\right\} \text{(OP-II)}.$$

From the assumption $c_i < c_{r,i}$ for all i it naturally follows that before any TU will be rented all own TU's must be allocated. Thus (OP-II) makes sense only if $\sum\limits_{i=1}^{M} n_i^* > K$, which we will assume in the following. At first we introduce for each i the differences

$$\Delta_n k_i(n_i,r_i) := k_i(n_i+1,r_i)-k_i(n_i,r_i) \quad \text{and} \quad \Delta_r k_i(n_i,r_i) := k_i(n_i,r_i+1)-k_i(n_i,r_i).$$

From (3.8) follows for all i that

$$\Delta_n k_i(n_i,r_i) = c_i + \begin{cases} -p_i\cdot Q, & (n_i+r_i+1)\cdot Q < d_i; \\ h_i\cdot Q-(h_i+p_i)\cdot[d_i-(n_i+r_i)\cdot Q], & (n_i+r_i)\cdot Q \le d_i < (n_i+r_i+1)\cdot Q; \\ h_i\cdot Q, & d_i < (n_i+r_i)\cdot Q \end{cases}$$

and

$$\Delta_r k_i(n_i,r_i) = \Delta_n k_i(n_i,r_i) + c_{r,i} - c_i.$$

At first sight we can not apply Marginal Analysis to solve (OP-II) because of the functions $k_i(\cdot,\cdot)$ depend now on two non-negative integer variables. But it is possible to transform these functions in such a way that they will depend on a single variable only. To realise that we argue as follows. Obviously, for the optimal solution $(\mathbf{n}^{(II)},\mathbf{r}^*)$ of (OP-II) holds $n_i^{(II)}+r_i^* \le n_i^*, i=1,\ldots,M$, where n_i^* is from (3.5). In case of $K=0$ it holds $\mathbf{n}^{(II)}=(0,\ldots,0)$ and $\mathbf{r}^*\le\mathbf{n}^*$. Since the number of rented TU's is not limited we can start our procedure to solve (OP-II) with $\mathbf{r}=\mathbf{n}^*$. The corresponding

cost for retailer i are

$$k_i(0,n_i^*) = c_{r,i} \cdot n_i^* + p_i(d_i - n_i^* \cdot Q)^+ + h_i(n_i^* \cdot Q - d_i)^+.$$

If we now replace n_i rented TU's by own TU's we get cost of

$$k_i(n_i, n_i^* - n_i) = c_i \cdot n_i + c_{r,i} \cdot n_i^* + p_i(d_i - n_i^* \cdot Q)^+ + h_i(n_i^* \cdot Q - d_i)^+$$

or

$$k_i(n_i, n_i^* - n_i) = n_i \cdot (c_i - c_{r,i}) + k_i(0,n_i^*) < k_i(0,n_i^*).$$

Consequently, since $k_i(0,n_i^*)$ is a constant and $c_i \leq c_{r,i}$ the cost saving

$$k_i(0,n_i^*) - k_i(n_i, n_i^* - n_i) = n_i \cdot (c_{r,i} - c_i)$$

replacing n_i rented TU's by own TU's is a linear, strictly increasing function of $n_i, n_i \leq n_i^*$. Thus to maximize the gain from replacing rented TU's by the available K own TU's we can directly apply Marginal Analysis if we put $f_i(n_i) = n_i \cdot (c_{r,i} - c_i), i = 1, \ldots, M$. Again the solution process can be sped up using the linearity property of $f_i(n_i)$. For this we number the retailers such that $c_{r,1} - c_1 \geq c_{r,2} - c_2 \geq \cdots \geq c_{r,M} - c_M > 0$. Then it is obvious that for retailer 1 the number of replaced rented TU's by own TU's is equal to $min(n_1^*, K)$. If $K \geq n_1^*$ then retailer 2 replaces $min(n_2^*, K - n_1^*)$ rented TU's. Continuing that replacement process until all K TU's are allocated the process stops at a retailer $j \leq M$. Now we must consider two cases - at retailer j all rented TU's can be replaced and not all rented TU's can be replaced. In the first case we have $\mathbf{n} = (n_1^*, \ldots, n_j^*, 0, \ldots, 0)$ and $\mathbf{r} = (0, \ldots, 0, n_{j+1}^*, \ldots, n_M^*)$ as up-to now calculated solution of (OP-II). To finish the solution process it remains to calculate for retailers $i = j+1, \ldots, M$ the optimal number r_i^* of rented TU's. It is obvious that the optimality condition for each i is similar to (3.5), i.e., r_i^* can be calculated from

$$r_i^* = argmin\{r : (r+1) \geq \frac{d_i}{Q} + \frac{p_i - \frac{c_{r,i}}{Q}}{h_i + p_i}\} = argmax\{r : r \leq \frac{d_i}{Q} + \frac{p_i - \frac{c_{r,i}}{Q}}{h_i + p_i}\}. \tag{3.10}$$

We remark that from (3.5) and (3.10) follows that $r_i^* \leq n_i^*$ for $i = 1, \ldots, M$.

In the second case retailer j cannot replace all rented TU's by own TU's. Thus the up-to now calculated solution is $\mathbf{n} = (n_1^*, \ldots, n_{j-1}^*, n_j, 0, \ldots, 0)$ and $\mathbf{r} = (0, \ldots, 0, r_j, n_{j+1}^*, \ldots, n_M^*)$ with $0 \leq n_j \leq n_j^*$ and $r_j = n_j^* - n_j \geq 0$. For a retailer j with own and rented TU's, whereby the number n_j of own TU's is fixed, the optimal number $r_j(n_j)$ of rented TU's has also to fulfil the optimality condition

(3.10), i.e.,

$$\begin{aligned} r_j(n_j) &= argmax\{r : r + n_j \leq \frac{d_j}{Q} + \frac{p_j - c_{r,j}/Q}{h_j + p_j}\} \\ &= argmax\{r : r \leq \frac{d_j - n_j \cdot Q}{Q} + \frac{p_j - c_{r,j}/Q}{h_j + p_j}\}. \end{aligned} \tag{3.11}$$

Again it remains to calculate for retailers $i = j+1, \ldots, M$ from (3.10) the optimal numbers r_i^*. Summarizing these considerations we get an algorithm similar to algorithm OAP for (OP-I).

Algorithm *Optimal Allocation Procedure (OAP-II)* (to solve (OP-II) in case $\sum_{i=1}^{M} (n_i^*) \geq K$).

INPUT K; Q; $d_i, i = 1 \ldots M$.

1. Preliminaries:
   Calculate n_i^* from (3.5), $i = 1, \ldots, M$.
2. Initialization:
   $n_i := 0, r_i := n_i^*, i = 1, \ldots, M; nsum := 0$.
3. Sorting:
   Define a permutation $i = (i_1, \ldots, i_M)$ of integers 1 to M such that
   $c_{r,i_1} - c_{i_1} \geq c_{r,i_2} - c_{i_2} \geq \cdots \geq c_{r,i_M} - c_{i_M}$
4. Iteration:
   WHILE $nsum < K$ DO
   a) Allocation-step:
      $n_{i_1} := \min\left(n_{i_1}^*, K - nsum\right); r_{i_1} := r_{i_1} - n_{i_1}; nsum := nsum + n_{i_1}$.
   b) Resorting-step:
      Delete i_1 from the permutation and renumber the remaining elements.
5. Final step:
   FOR $i := 1$ to M DO
   IF $r_i \geq 0$ THEN $r_i := r_i^{(II)}$, where $r_i^{(II)}$ is calculated from (3.10) if $n_i = 0$ and from (3.11) otherwise.

OUTPUT Optimal allocation vector $\mathbf{n}^{(II)}$ and $\mathbf{r}^{(II)}$.

We want to remark that from the algorithms OAP and OAP-II follows that the two optimization problems (OP-I) and (OP-II) can have very different solutions. Nevertheless they have a common structural property - there exists at most a single retailer (retailer j in OAP-II) with two different transportation modes (if we assume for (OP-I) that no transport is a special mode). Of cause it

is no problem, having the optimal allocation vectors, to calculate the values of the corresponding goal functions.

Let us finally consider a simple numerical example.

Example 3.1 We assume $M = 10$ and $Q = 10$. The cost parameters and two demand vectors are given in Table 1. Table 1 contains also the values for $p_i \cdot Q - c_i$ and $c_{r,i} - c_i$ with the corresponding order places in i. The last two columns contain the optimal TU-values, calculated from (3.5) respectively (3.10). We apply now OAP and OAP-II to the two demand vectors.

i	1	2	3	4	5	6	7	8	9	10
c_i	1	2	3	4	5	6	5	4	3	2
$c_{r,i}$	10	10	10	10	10	10	10	10	10	10
h_i	1	1	1	1	1	1	1	1	1	1
p_i	3	4	5	6	7	8	9	8	7	6
$d_i(1)$	12	24	36	48	60	72	60	48	36	24
$d_i(2)$	100	150	80	30	5	20	200	60	40	90
$p_iQ - c_i$	29	38	47	56	65	74	85	76	67	58
i	10	9	8	7	5	3	1	2	4	6
$c_{r,i} - c_i$	9	8	7	6	5	4	5	6	7	8
i	1	2	4	6	8	10	9	7	5	3
n_i^*	1	3	4	5	6	8	6	5	4	3
r_i^*	1	3	4	5	6	7	6	5	4	3

Table 1: Data for Example 3.1

(i) Demand vector $\mathbf{d}^{(1)}$.

Let us assume $K = 25$. First we solve (OP-I) applying OAP.

1. Preliminaries: For the present $\mathbf{d}^{(1)}$ we get $\mathbf{n}' = (1,2,3,4,6,7,6,4,3,2)$.

2. Initialization: $n_i := 0, i = 1,\ldots,M; nsum := 0$.

3. Sorting: $\mathbf{i} = (7,8,6,9,5,10,4,3,2,1)$.

4. Iteration:

(1) $nsum < 25 \rightarrow$ a) $n_7 = 0$ and $n_7' > 0$ gives $n_7 = 6, nsum = 6$.
b) $\Delta k_7(6) = c_7 + h_7 Q > 0$
gives $\mathbf{i} = (8,6,9,5,10,4,3,2,1,7)$.

(2) $nsum = 6 < 25 \rightarrow$ a) $n_8 = 0$ and $n_8' > 0$ gives $n_8 = 4, nsum = 10$.
b) $\Delta k_8(4) = c_8 + h_8 Q - (h_8 + p_8)(d_8 - 4Q) = -58$
gives $\mathbf{i} = (6,9,5,8,10,4,3,2,1,7)$.

(3) $nsum = 10 < 25 \rightarrow$ a) $n_6 = 0$ and $n_6' > 0$ gives $n_6 = 7, nsum = 17$.
b) $\Delta k_6(7) = c_6 + h_6 Q - (h_6 + p_6)(d_6 - 7Q) = -2$
gives $\mathbf{i} = (9,5,8,10,4,3,2,1,6,7)$.

(4) $nsum = 17 < 25 \rightarrow$ a) $n_9 = 0$ and $n_9' > 0$ gives $n_9 = 3, nsum = 20$.
b) $\Delta k_9(3) = c_9 + h_9 Q - (h_9 + p_9)(d_9 - 3Q) = -35$
gives $\mathbf{i} = (5,8,10,4,3,2,9,1,6,7)$.

(5) $nsum = 20 < 25 \rightarrow$ a) $n_5 = 0$ and $n_5' > 0$ gives $n_5 = 5, nsum = 25$.
b) $\Delta k_5(5) = c_5 - p_5 Q = -65$
gives $\mathbf{i} = (8,5,10,4,3,2,9,1,6,7)$.

Output: $\mathbf{n}^{(I)} = (0,0,0,0,5,7,6,4,3,0)$ with $k(\mathbf{n}^{(I)}, \mathbf{d}^{(1)}) = 1058$.

Now we allow renting of TU's, i.e., we apply OAP-II.

1. Preliminaries: From (3.5) we get $\mathbf{n}^* = (1,3,4,5,6,8,6,5,4,3)$.

2. Initialization: $n_i := 0, r_i := n_i^*, i = 1,\ldots,M; nsum := 0$.

3. Sorting: $\mathbf{i} = (1,2,10,3,9,4,8,5,7,6)$.

4. Iteration:

(1) $nsum = 0 < 25 \rightarrow$ a) $n_1 = 1, r_1 = 0, nsum = 1$. b) $\mathbf{i} = (2,10,3,9,4,8,5,7,6)$.
(2) $nsum = 1 < 25 \rightarrow$ a) $n_2 = 3, r_2 = 0, nsum = 4$. b) $\mathbf{i} = (10,3,9,4,8,5,7,6)$.
(3) $nsum = 4 < 25 \rightarrow$ a) $n_{10} = 3, r_{10} = 0, nsum = 7$. b) $\mathbf{i} = (3,9,4,8,5,7,6)$.
(4) $nsum = 7 < 25 \rightarrow$ a) $n_3 = 4, r_3 = 0, nsum = 11$. b) $\mathbf{i} = (9,4,8,5,7,6)$.
(5) $nsum = 11 < 25 \rightarrow$ a) $n_9 = 4, r_9 = 0, nsum = 15$. b) $\mathbf{i} = (4,8,5,7,6)$.
(6) $nsum = 15 < 25 \rightarrow$ a) $n_4 = 5, r_4 = 0, nsum = 20$. b) $\mathbf{i} = (8,5,7,6)$.
(7) $nsum = 20 < 25 \rightarrow$ a) $n_8 = 5, r_8 = 0, nsum = 25$. b) $\mathbf{i} = (5,7,6)$.

5. Final step:

$\mathbf{n}$ = (1, 3, 4, 5, 0, 0, 0, 5, 4, 3) $\quad$ $\mathbf{r}$ = (0, 0, 0, 0, 6, 8, 6, 0 ,0, 0)
$\mathbf{r}^*$ = (1, 3, 4, 5, 6, 7, 6, 5, 4, 3)
$\mathbf{n}^{(II)}$ = (1, 3, 4, 5, 0, 0, 0, 5, 4, 3) and $\mathbf{r}^{(II)}$ = (0, 0, 0, 0, 6, 7, 6, 0, 0, 0).

Output: $\mathbf{n}^{(II)}, \mathbf{r}^{(II)}$ and $k(\mathbf{n}^{(II)}, \mathbf{r}^{(II)}, \mathbf{d}^{(1)}) = 313$, which is 29,58% of $k(\mathbf{n}^{(I)}, \mathbf{d}^{(1)})$.

(ii) Demand vector $d^{(2)}$.

Let us assume now $K = 50$. Again we solve first (OP-I) applying OAP.

1. Preliminaries: For $d^{(2)}$ we get $\mathbf{n}' = (10,15,8,3,0,2,20,6,4,9)$.

2. Initialization: $n_i := 0, i = 1,\ldots,M; nsum := 0$.

3. Sorting: $\mathbf{i} = (7,8,6,9,5,10,4,3,2,1)$.

4. Iteration:

(1) $nsum = 0 < 50 \rightarrow$ a) $n_7 = 0$ and $n'_7 > 0$ gives $n_7 = 20, nsum = 20$.
b) $\Delta k_7(20) = c_7 + h_7 Q = 15$.
gives $\mathbf{i} = (8,6,9,5,10,4,3,2,1,7)$.

(2) $nsum = 20 < 50 \rightarrow$ a) $n_8 = 0$ and $n'_8 > 0$ gives $n_8 = 6, nsum = 26$.
b) $\Delta k_8(6) = c_8 + h_8 Q = 14$
gives $\mathbf{i} = (6,9,5,10,4,3,2,1,8,7)$.

(3) $nsum = 26 < 50 \rightarrow$ a) $n_6 = 0$ and $n'_6 > 0$ gives $n_6 = 2, nsum = 28$.
b) $\Delta k_6(2) = c_6 + h_6 Q = 16$
gives $\mathbf{i} = (9,5,10,4,3,2,1,8,7,6)$.

(4) $nsum = 28 < 50 \rightarrow$ a) $n_9 = 0$ and $n'_9 > 0$ gives $n_9 = 4, nsum = 32$.
b) $\Delta k_9(4) = c_9 + h_9 Q = 13$
gives $\mathbf{i} = (5,10,4,3,2,1,9,8,7,6)$.

(5) $nsum = 32 < 50 \rightarrow$ a) $n_5 = 0$ and $n'_5 = 0$ gives $n_5 = 1, nsum = 33$.
b) $\Delta k_5(1) = c_5 + h_5 Q = 15$
gives $\mathbf{i} = (10,4,3,2,1,9,8,7,5,6)$.

(6) $nsum = 33 < 50 \rightarrow$ a) $n_{10} = 0$ and $n'_{10} > 0$ gives $n_{10} = 9, nsum = 42$.
b) $\Delta k_{10}(9) = c_9 + h_9 Q = 12$
gives $\mathbf{i} = (4,3,2,1,10,9,8,7,5,6)$.

(7) $nsum = 42 < 50 \rightarrow$ a) $n_4 = 0$ and $n'_4 > 0$ gives $n_4 = 3, nsum = 45$.
b) $\Delta k_4(3) = c_4 + h_4 Q = 14$
gives $\mathbf{i} = (3,2,1,10,9,8,4,7,5,6)$.

(8) $nsum = 45 < 50 \rightarrow$ a) $n_3 = 0$ and $n'_3 > 0$ gives $n_3 = 5, nsum = 50$.
b) $\Delta k_3(5) = c_3 - p_3 Q = -47$
s gives $\mathbf{i} = (3,2,1,10,9,8,4,7,5,6)$.

Output: $\mathbf{n}^{(I)} = (0,0,5,3,0,2,20,6,4,9)$ with $k(\mathbf{n}^{(I)}, \mathbf{d}^{(1)}) = 1278$.

The application of OAP-II gives following results.

1. Preliminaries: From (3.5) we get $\mathbf{n}^* = (10,15,8,3,1,2,20,6,4,9)$.

2. Initialization: $n_i := 0, r_i := n_i^*, i = 1, \ldots, M; nsum := 0$.

3. Sorting: $\mathbf{i} = (1,2,10,3,9,4,8,5,7,6)$.

4. Iteration:

(1) $nsum = 0 < 50 \rightarrow$ a) $n_1 = 10, r_1 = 0, nsum = 10.$ b) $\mathbf{i} = (2,10,3,9,4,8,5,7,6)$.
(2) $nsum = 10 < 50 \rightarrow$ a) $n_2 = 15, r_2 = 0, nsum = 25.$ b) $\mathbf{i} = (10,3,9,4,8,5,7,6)$.
(3) $nsum = 25 < 50 \rightarrow$ a) $n_{10} = 9, r_{10} = 0, nsum = 34.$ b) $\mathbf{i} = (3,9,4,8,5,7,6)$.
(4) $nsum = 34 < 50 \rightarrow$ a) $n_3 = 8, r_3 = 0, nsum = 42.$ b) $\mathbf{i} = (9,4,8,5,7,6)$.
(5) $nsum = 42 < 50 \rightarrow$ a) $n_9 = 4, r_9 = 0, nsum = 46.$ b) $\mathbf{i} = (4,8,5,7,6)$.
(6) $nsum = 46 < 50 \rightarrow$ a) $n_4 = 3, r_4 = 0, nsum = 49.$ b) $\mathbf{i} = (8,5,7,6)$.
(7) $nsum = 49 < 50 \rightarrow$ a) $n_8 = 1, r_8 = 5, nsum = 50.$ b) $\mathbf{i} = (5,7,6)$.

5. Final step:

$\mathbf{n}$ = (10, 15, 8, 3, 0, 0, 0, 1, 4, 9) $\mathbf{r}$ = (0, 0, 0, 0, 1, 2, 20, 5 ,0, 0)
$\mathbf{r}^*$ = (10, 15, 8, 3, 1, 2, 20, 6, 4, 9)
$\mathbf{n}^{(II)}$ = (10, 15, 8, 3, 0, 0, 0, 1, 4, 9) and $\mathbf{r}^{(II)}$ = (0, 0, 0, 0, 1, 2, 20, 5 ,0, 0).

Output: $n^{(II)}, r^{(II)}$ and $k(\mathbf{n}^{(II)}, \mathbf{r}^{(II)}, \mathbf{d}^{(2)}) = 395$, which is $30,91\%$ of $k(\mathbf{n}^{(I)}, \mathbf{d}^{(2)})$.

The results for Example 3.1 show that the solutions of (OP-I) and (OP-II) in general are very different. That fact has some consequences for practice - if there exists the possibility to rent TU's the optimal solution differs considerably (dependent on the demand realisation) as well as in cost as in the allocation vector from the solution without renting. Furthermore, from the algorithm OAP-II follows that the solution for (OP-II) possesses an interesting property: There exists at most a single retailer with two different TU-types. The demand in all other retailers will be satisfied else by own TU's or by rented TU's. This in some sense facilitates the organisation of the transportation in reality.

In the present chapter we assumed a fixed number of own TU's. The fleet-sizing problem however looks for an optimal number of fleet units. This makes sense only for the multi-period situation. To formulate corresponding optimization problems we have to adapt the cost structure. For instance we have to introduce some cost for holding a TU as well as for an own TU not used during a period. Another problem is the assumption of deterministic known demand over the planning horizon. It must be expected however that the demand is stochastic. For such a situation queueing theory places as our disposal some approaches. We will briefly discuss that topic in the following chapter.

4 A queueing model

Up-to now we investigated discrete time models. In the present section we consider the continuous time case and some queueing models. Since analytical solutions are possible only under some simplifying assumptions we define the following basic model:

1. A single hub and M spokes or retailers.
2. The hub has an ample amount of a single product and a fleet of K identical TU's.
3. Spoke i generates a demand for a single TU in accordance with a Poisson process with parameter $\lambda_i > 0$, $i = 1, 2, ..., M$.
4. The time for a trip from the hub to spoke i, for unloading and the return trip to the hub is an exponentially distributed random variable (r.v.) with parameter $\mu_i > 0$.
5. All random variables are independent.
6. Transportation orders will be served in accordance with first-come-first-served (FCFS) policy.
7. If all TU's are on the trip arriving transportation orders will be queued.
8. Following cost parts are considered:

   $c > 0$ - fixed cost per time unit for one TU,

   $w > 0$ - waiting cost per time unit and waiting order,

   $c_s > 0$ - cost per time unit for a used TU.

Remark 4.1 We remark that the defined model is a $M/M/K/\infty$ queueing system with parameters λ and μ, where $\lambda = \sum_{i=1}^{M} \lambda_i$ and $E(\text{service time}) = \frac{1}{\mu} = \sum_{i=1}^{M} \frac{\lambda_i}{\lambda} \times \frac{1}{\mu_i}$. Thus the condition $K > \frac{\lambda}{\mu}$ is necessary and sufficient for the existence of a stationary regime.

We formulate a first queueing optimization problem (QOP-I) as follows:

Define such a number K^* of TU's, which minimizes the average long-run cost per time unit.

To solve (QOP-I) is equivalent to the definition of the cost-optimal number of servers. For a mathematical formulation of (QOP-I) above all we have to formalize the criterion function. Let $C(K)$ denote the average cost per time unit in the steady-state regime if there are K servers. With the assumed cost structure (Assumption 8 for the model) it holds that

$$C(K) = K \cdot c + Q(K) \cdot w + E[\text{number of busy servers}] \cdot c_s,$$

where $Q(K)$ denotes the average number of waiting clients (orders) in the $M/M/K/\infty$ queueing system with K servers. Since for such a system we have $E[\text{number of busy servers}] = \frac{\lambda}{\mu}$ we finally can write, using the abbreviation $a = \frac{\lambda}{\mu}$, that

$$C(K) = a \cdot c_s + K \cdot c + Q(K) \cdot w. \tag{4.1}$$

Having in mind that for the existence of steady-state regime S should be greater than a we write (QOP-I) as

$$\left.\begin{array}{ll} & \text{Minimize } C(K) \\ \text{s.t.} & \\ & K > \frac{\lambda}{\mu} \\ & K \in \mathbb{N} \end{array}\right\} \text{(QOP-I)}$$

Again, to define an optimization algorithm properties of $Q(K)$ with respect to K will be useful. Thus we consider some important performance measures for $M/M/K/\infty$ queueing systems. We remarked above that for existence of the steady-state regime the number K of servers must be greater than $a = \frac{\lambda}{\mu}$. For all $K > a$ we have following formulas for the different performance measures (see e.g. [Tri82]):

(i) For the steady-state probabilities p_k that k jobs resp. clients are in the system it holds

$$p_0 = \frac{1}{\sum_{k=0}^{K} \left(\frac{a^k}{k!}\right) + \frac{a^{K+1}}{K!(K-a)}} \text{ and } p_k = p_0 \times \begin{cases} \frac{a^k}{k!}, & k \le K \\ \frac{a^k}{K!K^{k-K}}, & k \ge K \end{cases} \quad k = 0, 1, 2, \ldots$$

(ii) The average number Q of waiting clients is given by $Q = p_0 \times \frac{a^K}{K!} \times K \times \frac{a}{(K-a)^2}$.

(iii) For the average waiting time W of a client holds

$$W = \lambda \cdot Q = \frac{a^k}{\mu \cdot (K-1)!(K-a)} \times \frac{1}{(K-a)\sum_{k=0}^{K-1} \frac{a^k}{k!} + \frac{a^K}{(K-1)!}} \tag{4.2}$$

(iv) The average number of clients in the system can be computed as $L = Q + a$.

(v) The average number of busy servers is equal to $E[\text{number of busy servers}] = a = \frac{\lambda}{\mu}$.

We remember that the criterion function for (QOP-I) is $C(K) = a \cdot c_s + K \cdot c + Q(K) \cdot w$. If we could prove that C(K) is integer-convex with respect to K, the design of an optimization algorithm is straightforward. Since $K \cdot c$ is a linear function we have to consider $Q(K)$ only. We notice that from equation $W = \lambda \cdot Q$ follows that performance measure $W(K)$ inherits all properties from $Q(K)$.

[DP77] proved the following

Theorem 4.1 In the $M/M/K/\infty$ system is the performance measure $Q(\cdot)$ a decreasing integer-convex functions of K for $K \geq a$.

With Theorem 4.1 the criterion function $C(K)$ is integer-convex with respect to the number K of servers. Now the validity of the following optimization algorithms lies on hand (where we use the notion $\lfloor a \rfloor$ for the greatest integer not exceeding a).

Algorithm *Optimal Number of Servers* (ONS)

1. Initialization: $K := \lfloor a \rfloor + 1$; $C_1 := C(K)$; $C_0 := MAX$.
2. WHILE ($C_1 < C_0$) DO

   BEGIN

   $K := K + 1$;

   $C_0 := C_1$;

   $C_1 := C(K)$

   END.
3. RETURN $K^* = K - 1$ and $C^* = C_0$.

Obviously the considered model and consequently (QOP-I) are very simple. Several generalizations are possible.

A) *With respect to the distributions.*
In case of arbitrary distribution functions for the inter-arrival and service times we get a $G/GI/K/\infty$ system. For such systems [Web80] proved

Theorem 4.2 For the $G/GI/K/\infty$ system the performance measures $Q(\cdot)$ and $W(\cdot)$ are decreasing integer-convex functions of K for $K \geq a$.

The problem however is that for $G/GI/K/\infty$ systems we have for Q and other performance measures only approximate formulas.

B) *With respect to the distribution laws of the retailers.*
We assume now that $A_i(\cdot)$ and $B_i(\cdot)$ denote the distribution function of the generation time for transportation orders in location i and the service time by the centre, respectively. We assume only that the first moments $m_1(A_i)$ and $m_1(B_i)$ are finite for all locations. If $\lambda_i = 1/m_1(A_i)$

denotes the arrival intensity of transportation orders from i then we get a $G/GI/K/\infty$ system by setting

$$\lambda = \sum_{i=1}^{M} \lambda_i, \quad A(t) = \sum_{i=1}^{M} \frac{\lambda_i}{\lambda} A_i(t), t \geq 0 \quad \text{and} \quad B(t) = \sum_{i=1}^{M} \frac{\lambda_i}{\lambda} B_i(t), t \geq 0$$

C) *With respect to the capacity of the order queue.*
We can assume that no backorders or only a finite number T of backorders is possible. Then we get the lost-case models $M/M/K/0$ or $G/GI/K/0$ respectively the finite models $M/M/K/T$ or $G/GI/K/T$.

D) *With respect to the optimization criterion.*
We can assume other criterion functions as well as other constraints.

Our first problem (QOP-I) is related to the fleet-sizing problem for a special continuous time model. Next we discuss a simple fleet-allocation problem, which is similar to that in Section 3 for the discrete time model. More concrete, we are looking for such a permanent allocation of a given number K of transportation units to retailers, which minimizes the sum of the average waiting times of all retailers. Let $W(n) = \sum_{i=1}^{M} W_i(n_i)$ denote the total average waiting time in the system for allocation vector $\mathbf{n} = (n_1, \ldots, n_M)$. Then we formulate (QOP-II) as

$$\left.\begin{array}{ll} & \text{Minimize } W(\mathbf{n}) \\ \text{s.t.} & \\ & \sum_{i=1}^{M} n_i \leq K \\ & n_i \geq \left\lfloor \frac{\lambda_i}{\mu_i} \right\rfloor + 1 \\ & n_i \in \mathbb{N}, i = 1, \ldots, M \end{array}\right\} \text{(QOP-II)}$$

This is a non-linear integer optimization problem with decreasing and convex functions $W_i(\cdot)$. Again we can apply Marginal Analysis and define the following algorithm.

Algorithm *Optimal Allocation of a Fleet* (OAF)

1. Initial solution $\mathbf{n}^{(0)} = (n_1^{(0)}, \ldots, n_M^{(0)})$ with $n_i^{(0)} \geq \left\lfloor \frac{\lambda_i}{\mu_i} \right\rfloor + 1$.
2. $k := 1$.
3. $\mathbf{n}^{(k)} = \mathbf{n}^{(k-1)} + \mathbf{e}_i$, where $\mathbf{e}_i = (0, 0, \ldots, 1, 0, \ldots, 0)$ and i that index, which maximizes $W_j(n_j^{(k-1)}) - W_j(n_j^{(k-1)} + 1)$.
4. Stop if $k = K - \left(M + \sum_{i=1}^{M} \left\lfloor \frac{\lambda_i}{\mu_i} \right\rfloor \right)$ Otherwise $k := k + 1$ and go to 3.

The (QOP-II) can be generalised to the consideration of cost:

$$\left.\begin{array}{ll} & \text{Minimize } W(\mathbf{n}) \\ \text{s.t.} & \\ & \sum_{i=1}^{M} c_i \cdot n_i \leq C \\ & n_i \geq \left\lfloor \frac{\lambda_i}{\mu_i} \right\rfloor + 1 \\ & n_i \in \mathbb{N}, i = 1, \ldots, M \end{array}\right\} \text{(QOP-III)}$$

.

where c_i denotes the cost for the allocation of a single server to location i.

For OP(III) we have simply to maximize in step 3 of the algorithm Marginal analysis the fraction

$$\frac{W_j(n_j^{(k-1)}) - W_j(n_j^{(k-1)} + 1)}{c_j}$$

.

The properties of that approach are similar to those from Chapter 3.

To demonstrate the algorithm we consider an example, taken from [Köc05].

Example 4.1 Let $M = 5$, $c = 20$ €/day, $w = 500$ €/day and $c_s = 100$ €/day. The arrival and service rates are given in Table 2. From the data in Table 2 and the formulas in Remark 4.1 we calculate $\lambda = 6 day^{-1}$, $\mu = 1 day^{-1}$. Since $a = \frac{\lambda}{\mu} = 6$, we need at least 7 servers respectively TU's. Table 3 contains the results of the numerical computations.

n	1	2	3	4	5
$\lambda_n \ [day^{-1}]$	0.3	1.2	0.6	2.4	1.5
$\mu_n \ [day^{-1}]$	0.5	1.0	0.3	2.0	1.25

Table 2: Arrival and service rates for Example 4.1

These results show the following:

1. The probability $p_0(K)$ that the system is empty is an increasing function of the server number K. From the formula for $p_0(K)$ we see that $\lim_{K\to\infty} p_0(K) = e^{-a} = e^{-6} = 0.00247875$.
2. The average number $Q(K)$ of waiting transportation orders in the steady-state regime is, as stated in Theorem 4.1, a convex function of the number K of TU's or servers.
3. The average cost per time unit in the steady-state regime if there are K servers, $C(K)$, is a convex function of the number K of TU's or servers.

K	$p_0(K)$	$Q(K)$	$C(K)$ [€/day]
7	0.00157878	3.682981	1 841.4904
8	0.00214238	1.070945	1 295.4726
9	0.00235231	0.391962	975.9810
10	0.00243174	0.151949	875.9744
11	0.00246166	0.059066	849.5332
12	0.00247273	0.022474	851.2371
13	0.00247670	0.008269	864.1346
14	0.00247808	0.002924	881.4618

Table 3: Numerical results for Example 4.1

4. The optimal number of TU's or servers is equal to $K^* = 11$. For the given values of the cost parameters and rates underestimation of K^* is more dangerous than overestimation.

5. With increasing K function $C(K)$ becomes a linear function with grade $c = 20$€/day. This follows from the fact that $Q(K)$ vanishes for $K \to \infty$.

5 General HAS models - a simulation optimisation approach

5.1 Simulation optimization

Solving an optimisation problem (OP) by analytical approaches the underlying system has to be reduced to an idealized model. However real life distribution networks are highly complex. Thus an idealisation will reduce the correctness of the results of the optimisation process.

Another way to optimize a given system is to simulate the behaviour of the investigated system under different configurations. By assessing the behaviour of the system we can get informations about it´s goodness under the given configuration. Using optimisation algorithms like the genetic algorithm or tabu search, new configurations can be produced, which may be better or lead to a better configuration.

This approach is called the simulation optimisation approach. The main improvement by using the simulation optimisation approach is that there is no need to simplify the real problem to an idealized model. Actually it is possible to represent most of the real life systems by simulation software. Furthermore simulation opens new chances in gaining system relevant data, since all implemented system details are traceable. Another advantage of simulation is that it is possible to deal with huge and even heterogeneous models.

In the following we will briefly discuss the application of simulation optimisation to more general HAS models.

5.2 Fundamentals

The simulation optimisation approach can be divided into two parts, the simulation and the optimisation. While simulation is used to asses a given solution, optimisation will provide new solutions. To solve a given OP we use the optimisation cycle shown in Figure 2.

By using knowledge about previous considered solutions an optimisation algorithm creates new solutions for the decision parameters of the problem, which has to be solved. Next the simulator is configured by the given data for the current solution(s) and he is started. After stopping the simulation the collected statistical data are assessed to gain information about the goodness of the considered solution(s). The optimisation software then will add the new solution to the knowledge base and compare it to the other solutions. To stop the optimisation cycle a stopping condition is assigned. Such conditions may be a given number of solutions, an improvement rate etc.

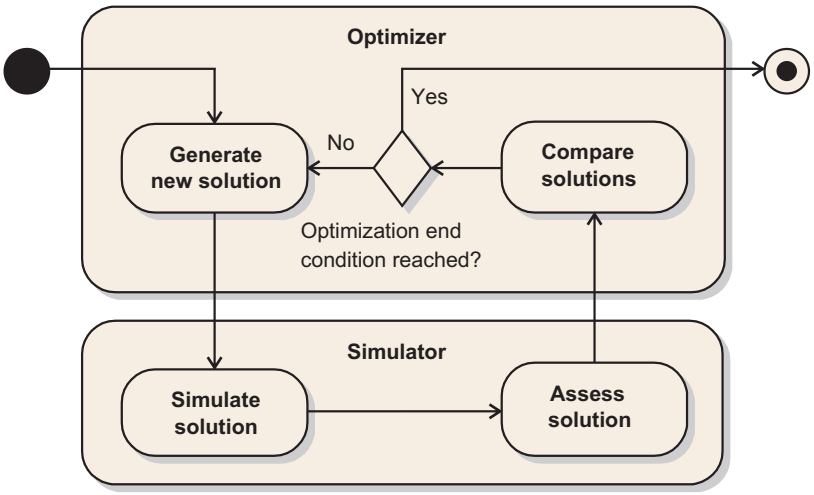


Figure 2: Scheme of simulation optimisation

5.3 CAOS

To realize the simulation optimisation approach we will use CAOS (Calculation Assessment Optimisation System), a software package, developed by Michael Kämpf at the professorship "Modelling and Simulation" at Chemnitz University of Technology (see [Käm05] and [Käm04]).

The idea of CAOS is to provide a software system, which separates the optimisation process from the optimisation problem. This means, it is only necessary to have information about the problem but not about the optimising algorithm(s). To solve an OP the user of CAOS has to build up a model of the system to which the problem is related. Furthermore he has to define the decision parameters and their domain. To solve the given OP an optimiser sets up the model with some special values for the decision parameters, and the simulation with these settings is started. As the result of the simulation we get at least one assessment value, on which the optimisation decisions are based. CAOS also allows multi criterion optimisation, this means it can also optimise models, which provide more than one assessment value. All assessment values are based on simulation data.

The whole environment of CAOS consists of:

- optimiser prototypes and some implementations, e. g., tabu search, genetic algorithms, etc., and also combinations of different optimisation algorithms,
- an analysis tool for tracing the optimisation process,
- model declaration language for building up general models,
- prototypes of simulators.

The CAOS software is extensible by implementing new optimisation algorithms and by defining types of optimisation problems. Important is another feature, that any model is defined by it´s general structure. CAOS provides an construction kit for any kind of system with the same structure. In case of the HAS structure this means that only the model elements hub and spoke must be declared. The number of spokes, the properties of the instances of model elements can be configured without reengineering the hub and spoke model. Once built up a model of a system this design of CAOS brings the advantage of short developing time, when creating a special realisation of a system.

In the following we describe the structure and behavior of the general HAS model

5.4 HAS Simulator

As written above we build up a construction kit for HAS structured simulators. At present it is possible to create HAS models with:

- a single hub,
- a nearly infinite number of homogeneous or heterogeneous spokes,

To realise various structures the simulation software contains the following model elements.

5.4.1 Model elements

product Currently there is only one product to be distributed. (Different products will differ in index i. Here we neglect this index because there is only one product.) All product quantities are measured in item units a.

inventory The amount of any inventory is measured in item units. While the capacity of the storage in a spoke is assumed to be finite the hub has an infinite amount of product to be distributed. For ordering reasons the inventory of a spoke is analysed by the inventory position r^{future}, which is equal to the inventory level r^{now} plus deliveries from open orders o minus unsatisfied demand d, i.e.,

$$r^{\text{future}} = r^{\text{now}} + o - d.$$

distance matrix Distances between all the locations (hub and spokes) are measured in distance units x.

transportation units It is possible to build a model with a given number of TU-classes. The classes differ with respect to the following adjustable values:

- transportation capacity Q (measured in item units a),
- speed v,
- transportation costs per product unit and distance unit $c^{\text{full}}_{a,x}$,
- transportation costs per empty capacity unit and distance unit $c^{\text{penalty}}_{a,x}$ ($c^{\text{penalty}}_{a,x}$ maybe less, equal or higher than $c^{\text{full}}_{a,x}$),
- fixed costs for loading and unloading c^{loading}, and finally
- rental costs for leasing a TU c^{rental}.

hub (central warehouse) Each model, which can be created, has a single hub. Furthermore the hub has a TU pool with different TU classes. The number of TU's in each class is a decision variable and can be optimised. In case an order of a retailer can not be satisfied, because of there is no TU in the TU pool, the order will be back ordered in a waiting queue. For this queue exist different service policies (see 5.4.2).

spoke (retailers) A model of a HAS structure can have a nearly infinite[2] number M of spokes but minimum one spoke is required.

Each spoke can have different, adjustable values for the:

- customers arriving process rate λ_k (deterministic or stochastic)
- customers demand d_k (deterministic or stochastic)
- customers waiting time limit t^{waiting}_k (deterministic or stochastic)
- customers service policy
- initial inventory level r
- ordering policy (see 5.4.2)

Initially each spoke has a given amount of product (maybe 0) in it's inventory. The spoke will release an order to the hub in accordance with its ordering policy.

For sold products there is a gain g_k. For the case a waiting customer leaves the queue because of he has reached his waiting time limit we assume a loss for not satisfied demand $c^{\text{shortage}}_{k,a}$. Finally, there are holding cost $c^{\text{holding}}_{k,a}$ for product in the storage.

5.4.2 Control policies

To control the product flow through the HAS distribution network we use some ordering and service policies.

The order policies are used by the spokes to refill their inventories. Following policy types are implemented:

[2] As much as the given hardware resources allow.

(s,S) If the level of the observed inventory position r is lower than bound s, an order up to level S is released.

(s,nQ) If the level of the observed inventory position r is lower than bound s, an order of n times the amount of Q, the lot size, is released.

Service policies are used by the spokes and the hub to satisfy waiting customers. In case of the hub the customers are the spokes. The hub does not provide the policy EDF, while the spokes do not have waiting time limits for their orders.

FIFO/FCFS (First In First Out / First Come First Serve) The waiting elements of the queue will be served in order they entered the queue.

LIFO/LCFS (Last In First Out / Last Come First Serve) The waiting elements of the queue will be served in reverse order they entered the queue.

EDF (Earliest Deadline First) The waiting elements of the queue will be served by their remaining waiting time, that means the element with the lowest waiting time will be served first (even if it's below zero!). Elements with the same left waiting time are served by FIFO.

SAN (Smallest Amount Next) The waiting elements of the queue will be served by the amount of their demand. The elements with the smallest amount will be served first.

BAN (Biggest Amount Next) The waiting elements of the queue will be served by the amount of their demand. The elements with the biggest amount will be served first.

Random The waiting elements of the queue will be served in random order.

5.4.3 Events - Activity diagrams

The sequence of events describes the evolution of the system in time. The event types, which are relevant for the considered hub-and-spoke system, are the arrival of a client with defined demand (new demand), the arrival of a TU with product at a spoke (TU delivery), and the return of a TU to the hub (TU return). In the activity diagrams below we show, which activities are related with the various events.

new demand This event causes all the dynamics in the hub and spoke structure. If it is occurred it will involve complex reactions.

After simulating demand and waiting time limit (they may be stochastic) the demand order is queued in the customers waiting list of the current spoke. To satisfy this demand the activities shown in Fig.4 have to be executed. The event "new demand" will always generate

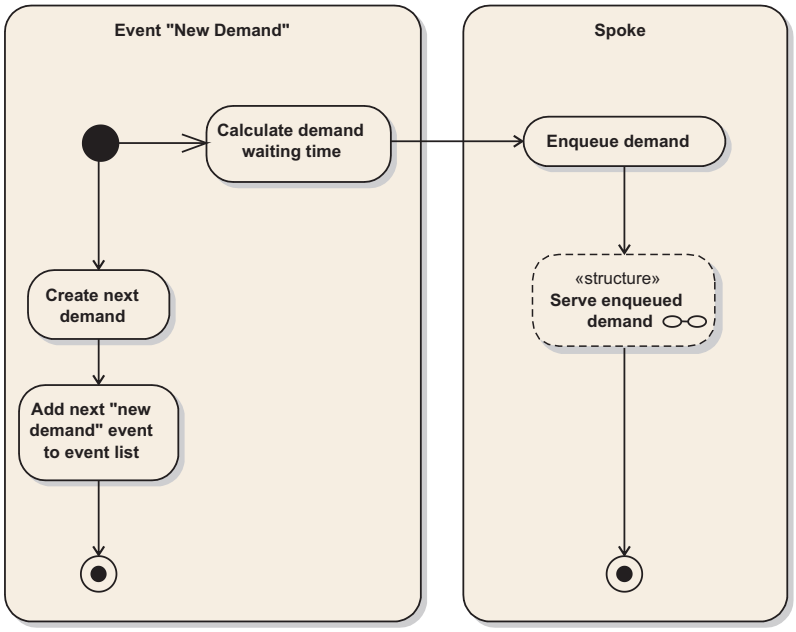


Figure 3: Activity Diagram: Event “new demand”

the time moment for the following event “new demand”. Thus the time evolvement of the model is kept running.

Figure 4 describes how the demand of any customer can be satisfied (if possible) and also what will happen if it can not be satisfied. Necessarily an order of products has to be placed. To deal with this the activities shown in Fig.5 must be realised.

Looking closer at the diagram not only tasks for running the simulation are executed, but also gain and cost are added to a cost manager. The cost manager will provide the assessment of the current HAS configuration. We will consider this part later.

When an order has been placed in the order queue of the spoke, transshipment decisions are necessarily to prove and execute (if possible). As shown in Fig.5 the transshipment is realized by the event types TU delivery and TU return. We will continue with these event types.

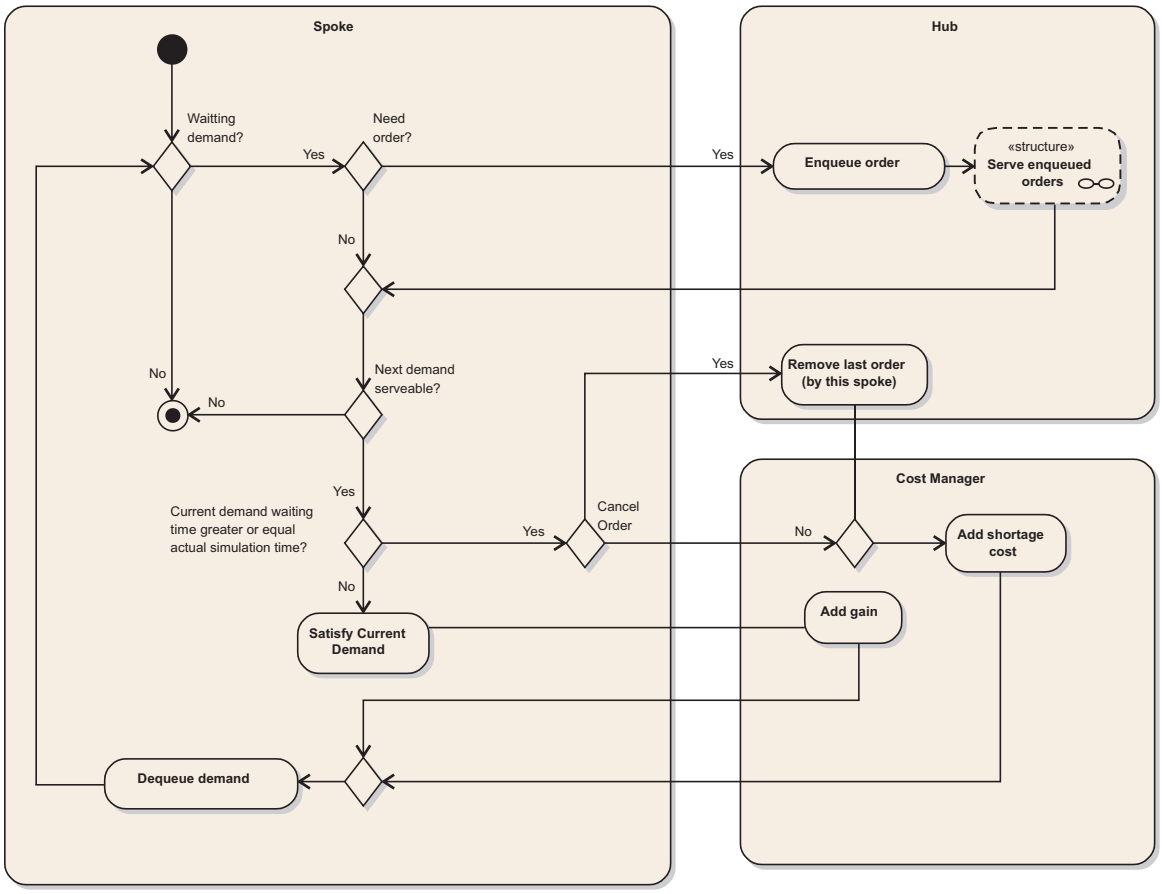


Figure 4: Activity Diagram: Serve Enqueued Demand

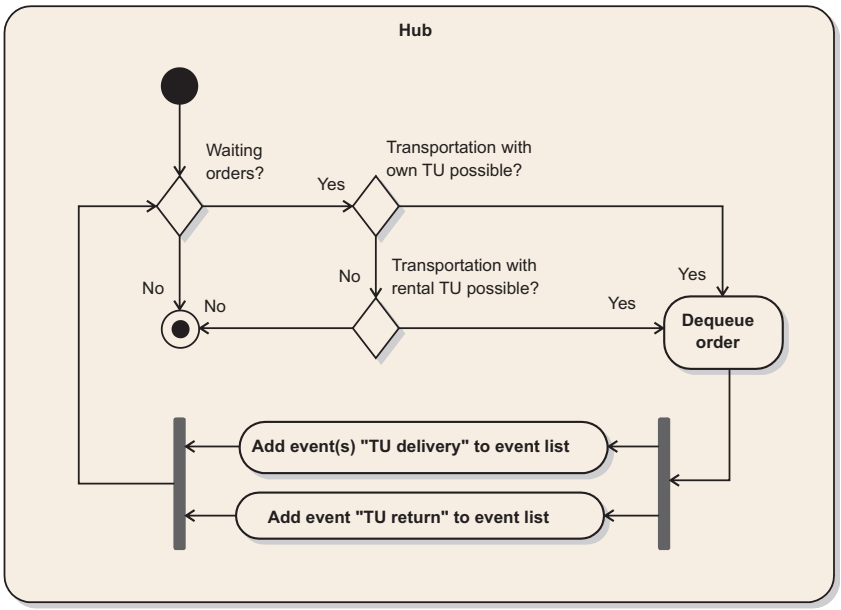


Figure 5: Activity Diagram: Serve Enqueued Orders

TU delivery This event will occur if a TU arrives at a spoke. It generates the activities shown in Fig.6.

Using one of the available service policies the hub selects the spoke fitting to this policy. According to the order(s) of the spoke the inventory will be filled. If the spoke has more than one open order and the amount of the first order would not use the whole transportation capacity of a TU, then this capacity will be used for further orders of the same spoke because of after the arrival of new product waiting customers can be served. To realise this the activities shown in Fig.4 will be executed.

Delivering products generates costs, which are added to the cost manager. The costs for loading and unloading are fixed costs, whereas the costs for transportation are proportional to the transported amount of product and the transport distance. The distance matrix gives the transport distance.

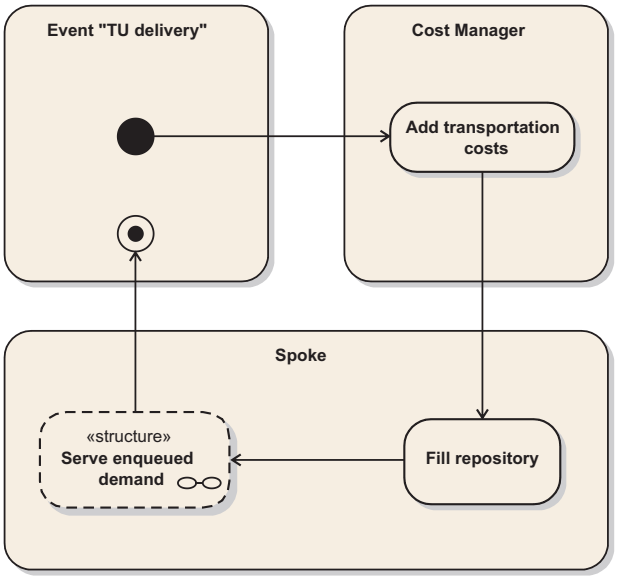


Figure 6: Activity Diagram: Event “TU delivery”

Refilling the inventory implies that customers may be served now. Therefor the activities shown in activity diagram Fig. 4 have to be executed.

TU return The event “TU return” describes what has to be done, when a TU returns to the spoke (see Fig.7). Obviously this includes the reactivation of the hub, so that transshipment decisions can be placed.

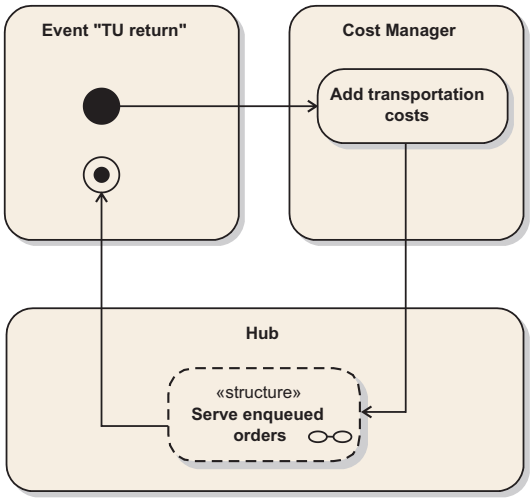


Figure 7: Activity Diagram: Event “TU return”

5.4.4 Assessment

As shown before there are several cost, which are added to a cost manager. Additionally to the up to now costs there are some other costs, which are collected over the simulation time. First of all there are holding costs $cost_{holding}$, which accrue for each spoke for positive inventory. The costs are added at time moments when the inventory level changes. These costs are assumed to be proportional to the storage time and the according amount of products. Furthermore costs for holding TU´s are added to the cost manager. These live time costs $cost_{TUlivetime}$ accrue on the the time a TU exists in the HAS. TU live time costs are collected for each TU.

Since the assessment of the current version of the simulator is based on a single criterion, all cost are combined to a total cost value $cost_{total}$, so that:

$$\begin{aligned} cost_{total} &= cost_{shortage} - gain \\ &+ cost_{transporter} + cost_{transportation} + cost_{TUlivetime} \\ &+ cost_{holding} \end{aligned}$$

The value of $cost_{total}$ provides information about the goodness of the current configuration of the HAS, comparing it to other configurations the configuration can be assessed. Obviously a higher value indicates a worse configuration and a HAS which shall pay has to have a negative value.

5.5 Examples

In the present paragraph we report on the simulation optimisation of two classes of hub-and-spoke systems - a single hub with 4 respectively 50 spokes. In all examples we assume the same demand process for all spokes. In detail we assume exponentially or normally distributed interarrival times of clients with exponentially respectively normally distributed demand per client. Similar distributions are assumed for the waiting time limits. Time is measured in time units, which may be one hour, one day, and so on. The other model parameters are as follows:

distances between hub and spokes	-	50 distance units;
TU capacity	-	100 item units;
lot size Q	-	10 item units;
TU speed	-	1 distance unit per time unit;
transportation cost $c_{a,x}^{\text{penalty}}$	-	0.1 per item unit and distance unit;
transportation cost $c_{a,x}^{\text{penalty}}$	-	2.0 per capacity unit and distance unit;
fixed loading/unloading cost c^{loading}	-	100;
service policy of the hub	-	FIFO;
service policy of the spokes	-	FIFO;
ordering policy of the spokes	-	(s, nQ);
initial inventory	-	zero;
gain g_k	-	12 per sold item unit;
shortage cost $c_{k,a}^{\text{shortage}}$	-	6 per item unit;
holding cost $c_{k,a}^{\text{holding}}$	-	0.12 per item unit and time unit.

The decision variables are the number T of TU's and for the spokes the order levels s and the number n of lot sizes. For the search for an optimal solution we restrict the possible values for T to the set 1, 2, ..., 10, for s to the set -120, -110, -100, ..., 120, and for n to the set 1, 2, ..., 20. We applied four optimisers - a Genetic algorithm, Tabu search, and two hybrid algorithms, which apply the Genetic algorithm and Tabu search in a parallel and a serial manner. The performance of a solution is estimated by the total cost, which accrue as the result of a single simulation run over 100 000 time units with a transition phase of 1 000 time units.

The graph in Fig.8 shows for the single-hub-four-spokes system with N(100, 10)-distributions, how the optimisation process evolves in time under the hybrid parallel algorithm. Similar pictures we have for all variants of the applied optimisers and all other examples (see Appendix B). Table 4 summarises the results for all four optimisers for the single-hub-four-spokes system. There are also given the results when the normal distributions are replaced by exponential distributions with parameter 0,1.

Optimization: Hybrid Parallel (N(100;10))

Solution number	1747			
Number of TU	2			
Spoke	1	2	3	4
Order Level	10	0	-10	0
Order Amount	130	60	200	60
Total Costs	−23.952520			

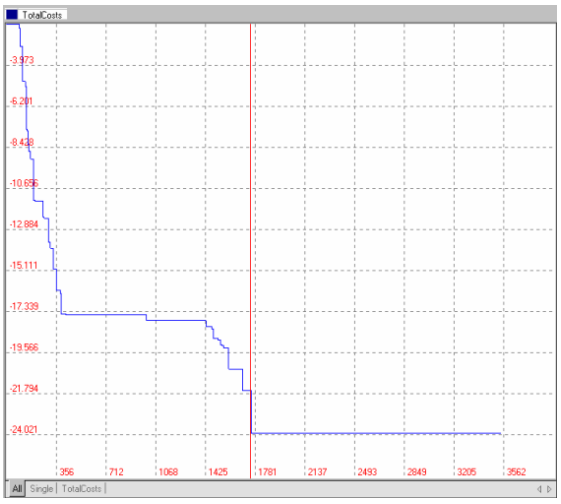


Figure 8: Simulation optimisation results for the four-spokes-single-hub system, N(100;10)-distributions

In the single-hub-fifty-spokes system the decision variable T can vary between 1 and 50. Table 5 consists the results, but without the policy parameters (see Appendix B for more information).

Considering the results in Tab. 4 and Tab. 5 we can formulate some conclusions.

1. The two hybrid optimisers and Tabu search outperform the Genetic algorithm. This was not to expect and gives a hint to the fact that the parameters of the Genetic algorithm may be bad adjusted to the considered problem. It is surprising that Tabu search works well.
2. The hybrid serial optimiser performs best, excluding the problem with 50 spokes and exponential distribution.
3. For the HAS with four spokes the optimisation process founds the best solution after maximum 2 500 considered solutions. For the much more complex problems with 50 spokes that maximal number increases up to about 18 000. Also the convergence to a good solution is considerably slower.
4. Very different solutions lead to similar total cost expectations. This can be a hint to the fact that our problem has many local minima with small cost differences.

To summarise we can say that the suggested simulation optimisation approach works well, but we

$N(100;\ 10)$ distribution						
	T	$(s,\ n)$				$c^{total\ cost}$
Hybrid parallel	2	(10, 13)	(0, 6)	(-10, 20)	(0, 6)	-23,95
Hybrid serial	9	(0, 6)	(0, 6)	(0, 6)	(0, 6)	-29,77
Genetic algorithm	4	(-30, 18)	(70, 7)	(0, 7)	(0, 11)	-15,00
Tabu search	9	(0, 6)	(0, 6)	(0, 6)	(0, 6)	-29,77
$EXP(0.1)$ distribution						
Hybrid parallel	6	(50, 9)	(40, 11)	(40, 9)	(40, 10)	-6,84
Hybrid serial	9	(40, 11)	(40, 11)	(40, 9)	(40, 10)	-6,86
Genetic algorithm	5	(20, 13)	(20, 13)	(20, 13)	(20, 13)	-4,39
Tabu search	10	(50, 9)	(50, 10)	(50, 8)	(50, 10)	-6,79

Table 4: Results of the simulation optimisation approach for the single hub and four spokes

	$N(100;\ 10)$		$EXP(0.1)$	
	T	$c^{total\ cost}$	T	$c^{total\ cost}$
Hybrid parallel	21	-260,66	50	-82,40
Hybrid serial	26	-359,75	30	-82,21
Genetic algorithm	18	-161,40	23	-57,18
Tabu search	23	-337,95	42	-85,82

Table 5: Results of the simulation optimisation approach for a single hub and 50 spokes

need more empirical material to improve the here-applied optimisers and to prove the above given conclusions.

Acknowledgement.
We are very grateful to Michael Kämpf for the support to use CAOS.

Marginal Analysis

FOX ([Fox66] has investigated for the problem

$$\begin{aligned} & \text{Maximise } f(\mathbf{n}) = \sum_{i=1}^{M} f_i(n_i) \\ \text{s.t.} \quad & \\ & C(\mathbf{n}) := \sum_{i=1}^{M} c_i \cdot n_i \leqslant C; \\ & n_i \in \mathbb{N}, i = 1, \ldots, M \end{aligned}$$

where $C \geq 0$ and $c : i \geq 0$, $i = 1,2,\ldots,M$, the following algorithm:

Algorithm *Marginal analysis* (MA):

1. Initialization:
   $r := 0; \mathbf{n}^r := (0,0,\ldots,0); C^r := 0.$
2. Search:
   WHILE $C^r < C$ DO

   BEGIN

   2.1. $r := r + 1.$

   2.2. Define index j, which maximizes $\frac{f_j(n_j^{r-1}+1)-f_j(n_j^{r-1})}{c_j}$.

   2.3. $\mathbf{n}^r := \mathbf{n}^{r-1} + \mathbf{e}_i.$

   2.4. $C^r := C^{r-1} + c_i.$

   END.
3. Output:
   $\mathbf{n}^{r-1}$ and C^{r-1}.

To describe properties of algorithm (MA) we call an allocation-vector $\mathbf{n}$ undominated if $\forall \mathbf{n}' \in \mathbb{N}^M$ from $f(\mathbf{n}') \geq f(\mathbf{n})$ follows $C(\mathbf{n}') \geq C(\mathbf{n})$. Obviously undominated vectors generate the Pareto-front of solutions. The main results of Fox (1966) are the following ones.

Property 1

If all $f_i(\cdot)$ are integer-concave and strictly increasing, $i = 1,2,\ldots,M$, then algorithm (MA) leads to undominated vectors $\mathbf{n}$.

Property 2

Let all $f_i(\cdot)$ be integer-concave and strictly increasing, $i = 1,2,\ldots,M$. Let $\mathbf{n}^1,\mathbf{n}^2,\ldots,\mathbf{n}^m$ denote the sequence of allocation-vectors generated by algorithm (MA), and let $\mathbf{n}^*$ denote the optimal allocation-vector. Then it holds:

$$
\begin{aligned}
(I) \quad & f(\mathbf{n}^{m-1}) \leq f(\mathbf{n}^*) < f(\mathbf{n}^m). \\
(II) \quad & C(\mathbf{n}^{m-1}) \leq C(\mathbf{n}^*) < C(\mathbf{n}^m). \\
(III) \quad & 0 < C(\mathbf{n}^m) - C(\mathbf{n}^m - 1) \leq \max_j c_j.
\end{aligned}
$$

Property 3

If, in addition to the conditions stated in Property 2, $c_1 = c_2 = c_M = c > 0$, then algorithm (MA) generates an optimal solution.

Remarks:

1. If Z denotes the set of all optimal allocations then $\mathbf{n}^{m-1}$ must not be an element of Z. However, (I) to (III) give some information on the quality of $\mathbf{n}^{m-1}$.
2. For $c_1 = c_2 = c_N = c > 0$ and C as an integer multiple of c the optimal value of the criterion $f(\mathbf{n}^*(C))$ is concave and strictly increasing with respect to C.
3. For a linear criterion $f(\mathbf{n}) = f_1 \cdot n_1 + \cdots + f_N \cdot n_N$ we have a version of the Knapsack-Problem.
4. Let the constraints be non-linear, i. e. $C(\mathbf{n}) = \sum_{i=1}^{N} c_i(n_i)$ with $c_i(\cdot)$ convex and strictly increasing for $i = 1,\ldots,M$. If in algorithm *Marginal analysis* Step 2.2 is replaced by

$$
\frac{f_j(n_j^{k-1}+1) - f_j(n_j^{k-1})}{c_j(n_j^{k-1}+1) - c_j(n_j^{k-1})},
$$

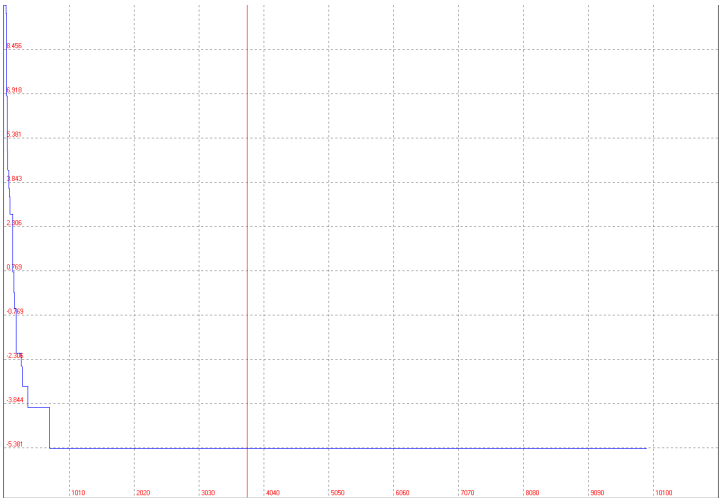
   then Property 1 holds also.

Simulation optimisation results

The present subsection contains the simulation optimisation results for the HAS with 4 and 50 spokes.

Optimization: Hybrid Parallel (Exp [0.1])

Solution number	565			
Number of TU	6			
Spoke	1	2	3	4
Order Level	50	40	40	40
Order Amount	90	110	90	100
Total Costs	−6.842771			



Optimization: Hybrid Seriell (Exp [0.1])

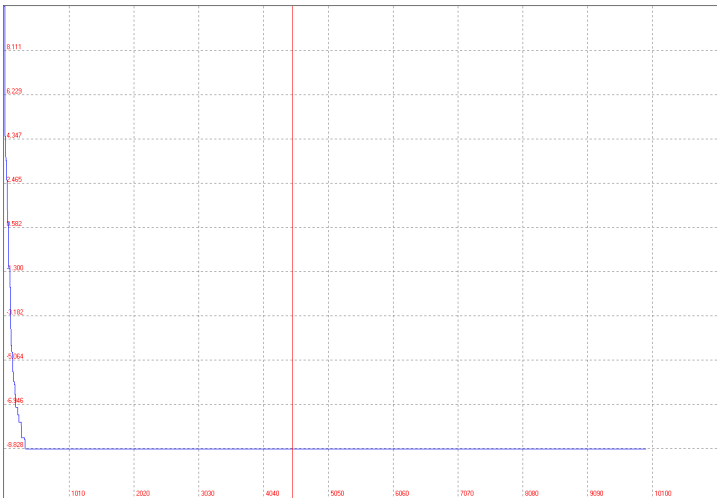
Solution number	1586			
Number of TU	9			
Spoke	1	2	3	4
Order Level	40	40	40	40
Order Amount	110	110	90	100
Total Costs	−6.864413			

Optimization: GA Generation (Exp [0.1])

Solution number	401			
Number of TU	5			
Spoke	1	2	3	4
Order Level	20	20	20	20
Order Amount	130	130	130	130
Total Costs	−4.388444			

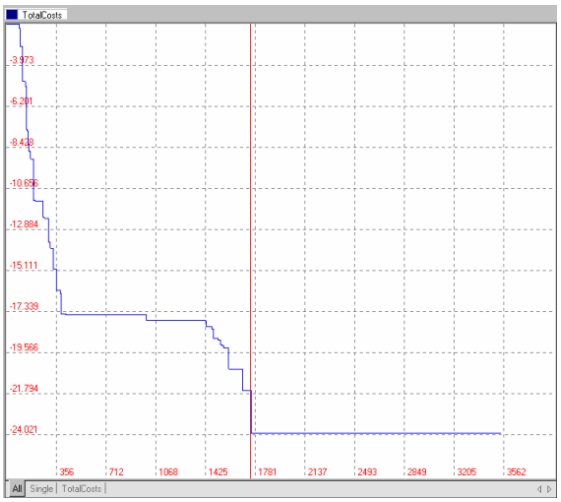
Optimization: Tabu Search (Exp [0.1])

Solution number	370			
Number of TU	10			
Spoke	1	2	3	4
Order Level	50	50	50	50
Order Amount	90	100	80	110
Total Costs	−6.786395			



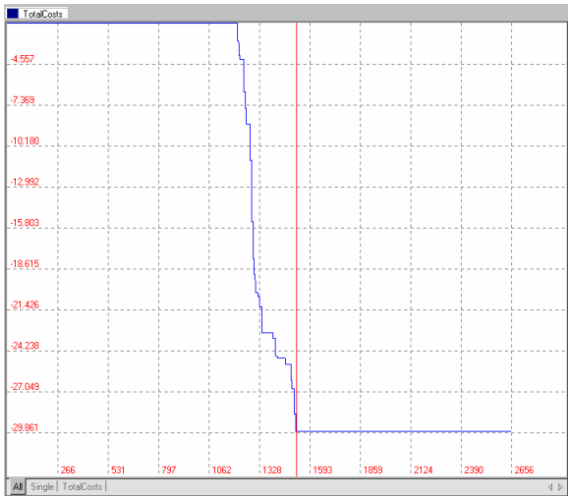
Optimization: Hybrid Parallel (N(100;10))

Solution number	1747			
Number of TU	2			
Spoke	1	2	3	4
Order Level	10	0	-10	0
Order Amount	130	60	200	60
Total Costs	−23.952520			



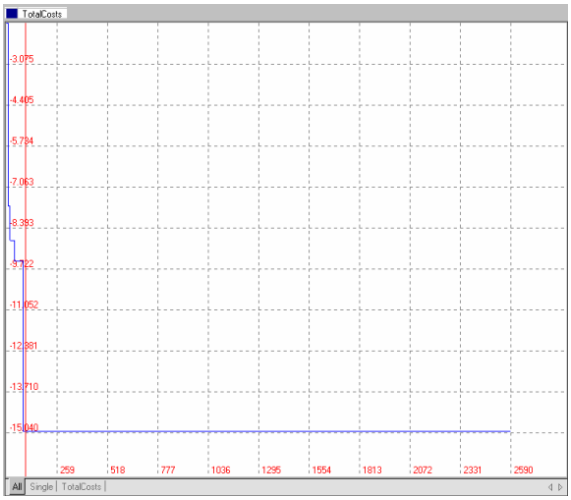
Optimization: Hybrid Seriell (N(100;10))

Solution number	1508			
Number of TU	9			
Spoke	1	2	3	4
Order Level	0	0	0	0
Order Amount	60	60	60	60
Total Costs	−29.774437			



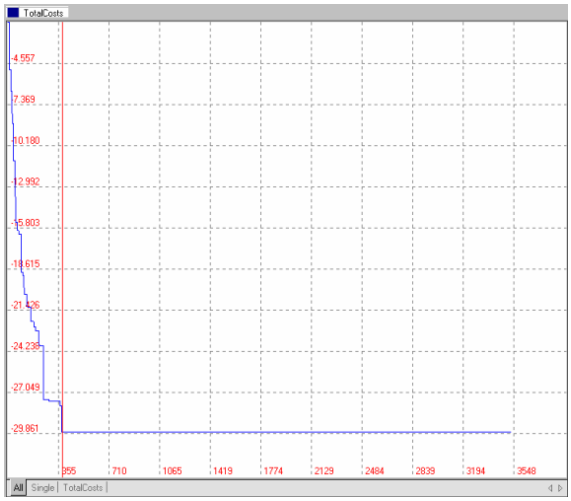
Optimization: GA Generation (N(100;10))

Solution number	85			
Number of TU	4			
Spoke	1	2	3	4
Order Level	-30	70	0	0
Order Amount	180	70	70	110
Total Costs	−14.998835			



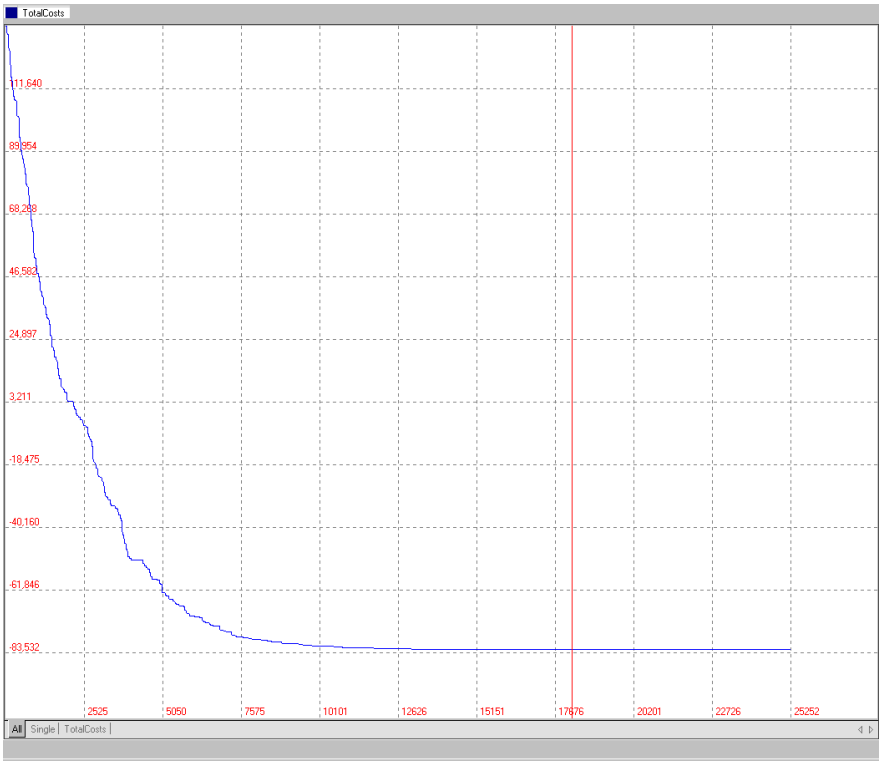
Optimization: Tabu Search (N(100;10))

Solution number	378			
Number of TU	9			
Spoke	1	2	3	4
Order Level	0	0	0	0
Order Amount	60	60	60	60
Total Costs	−29.774437			



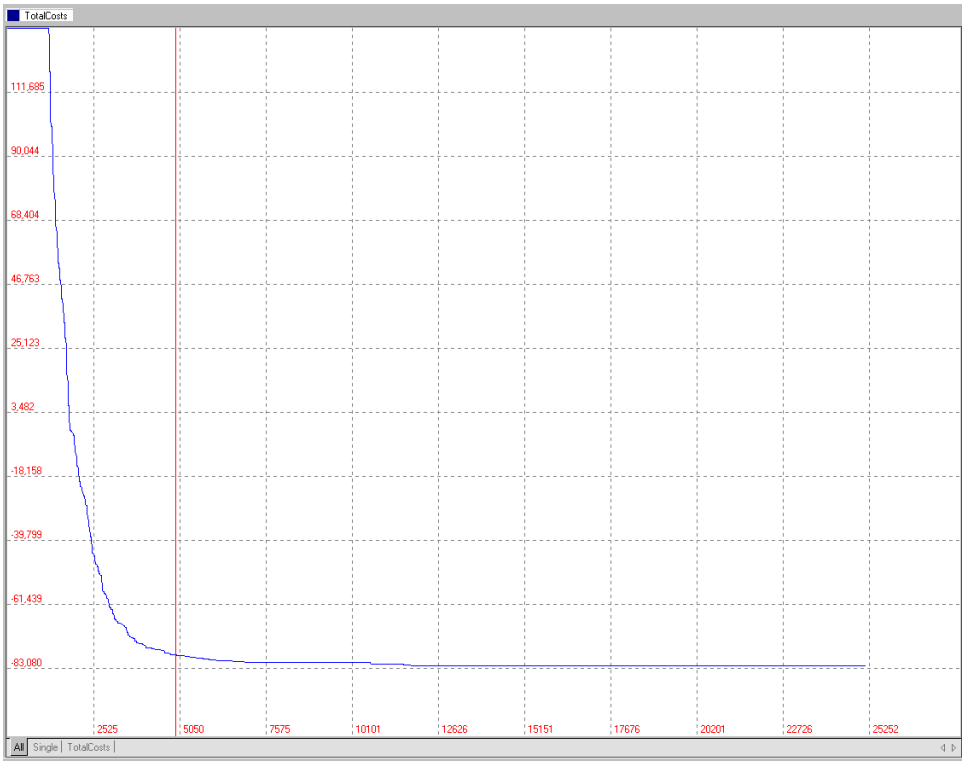
Optimization: Hybrid Parallel (Exp [0.1])

Solution number	18017									
Number of TU	50									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	50	50	40	40	50	50	50	50	50	40
Order Amount	100	90	110	100	90	90	90	100	100	120
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	40	40	50	40	50	50	40	40	50	40
Order Amount	110	100	100	100	120	100	100	110	110	100
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	40	50	50	40	40	50	50	30	40	40
Order Amount	110	100	90	100	110	100	110	110	90	100
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	40	50	50	50	50	50	50	50	50	30
Order Amount	110	110	100	90	100	90	80	100	90	120
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	40	40	40	50	40	50	50	40	40	50
Order Amount	90	100	110	90	90	80	100	110	110	100
Total Costs	−82.404133									



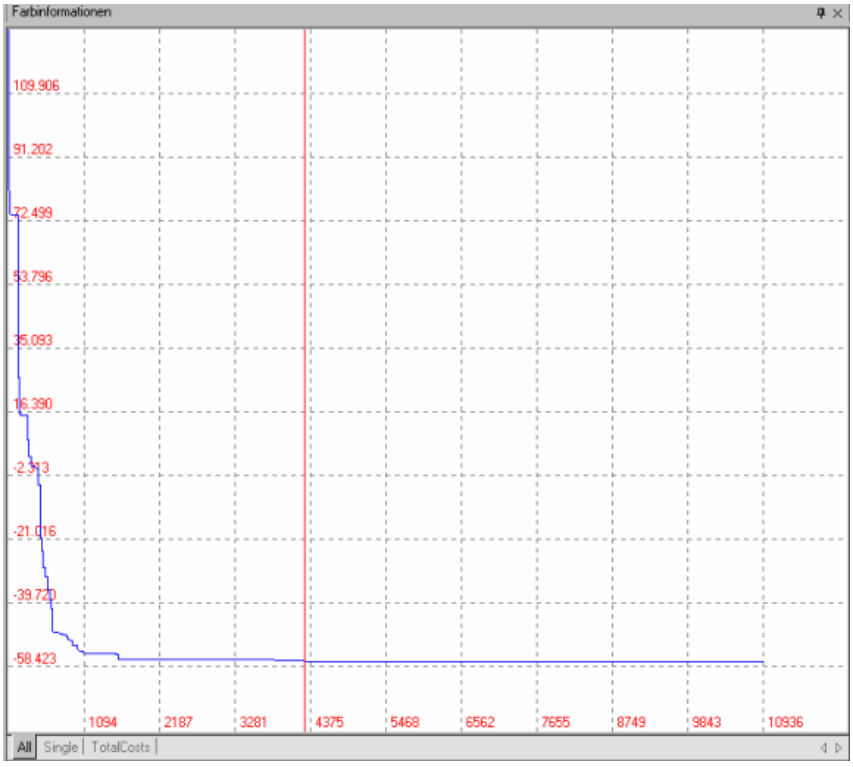
Optimization: Hybrid Seriell (Exp [0.1])

Solution number	13266									
Number of TU	30									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	50	50	40	40	40	50	40	50	50	50
Order Amount	100	90	110	100	110	100	110	100	100	120
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	40	50	40	40	50	40	40	50	50	40
Order Amount	110	100	110	100	120	110	100	90	110	90
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	40	50	40	40	40	50	50	50	30	40
Order Amount	110	100	100	100	110	100	110	100	120	100
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	40	40	50	50	50	40	50	50	40	40
Order Amount	110	110	100	100	100	110	90	100	100	110
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	50	40	40	50	40	30	50	40	40	40
Order Amount	110	100	110	90	90	110	100	90	110	90
Total Costs	−82.214317									



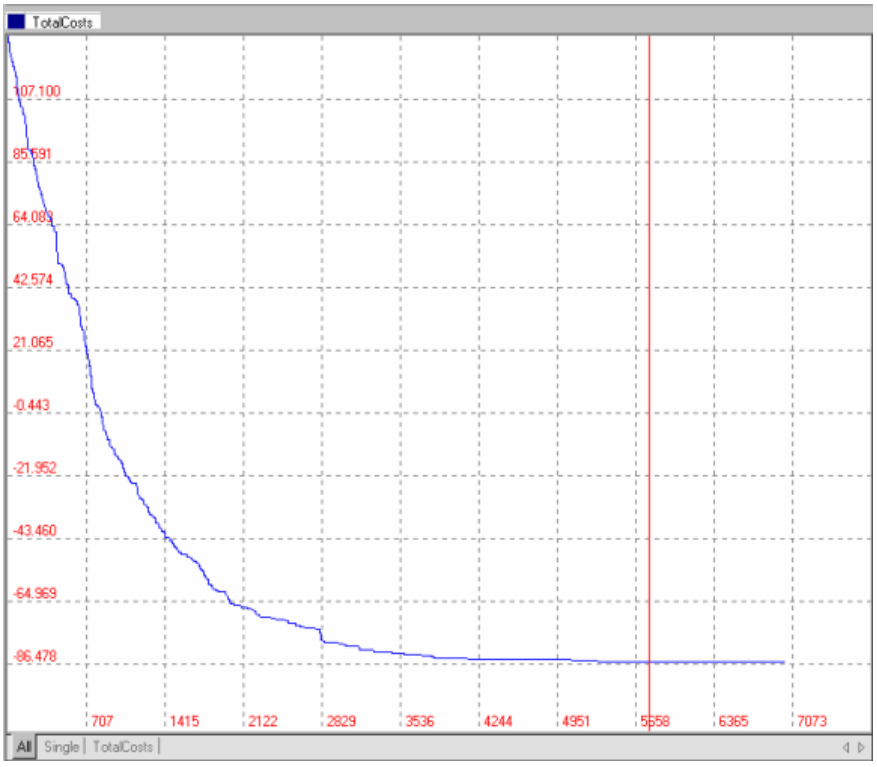
Optimization: GA Generation (Exp [0.1])

Solution number	4252									
Number of TU	23									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	30	30	30	30	30	30	30	30	30	30
Order Amount	140	140	130	130	130	130	130	140	140	140
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	30	30	30	30	30	30	30	30	30	30
Order Amount	130	130	130	130	130	140	140	140	130	130
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	30	30	30	30	30	20	20	30	30	30
Order Amount	140	140	140	140	130	130	130	140	140	140
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	30	30	30	30	30	30	30	30	30	30
Order Amount	140	130	130	130	130	130	130	140	140	140
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	30	40	40	40	50	40	40	40	30	30
Order Amount	140	140	140	140	140	140	140	140	140	140
Total Costs	−57.175815									



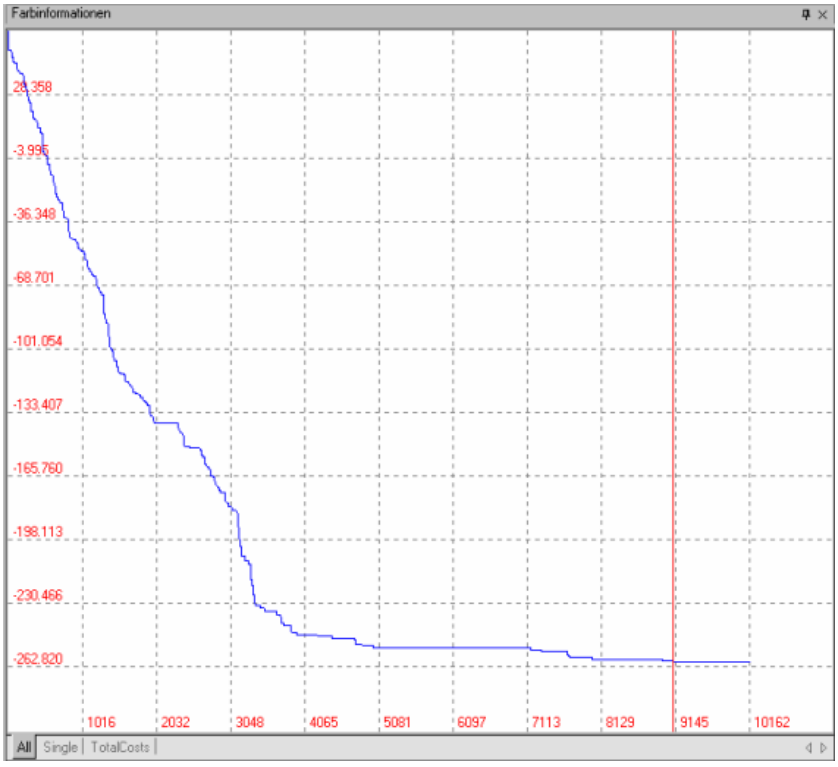
Optimization: Tabu Search (Exp [0.1])

Solution number	5764									
Number of TU	42									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	50	50	50	40	40	50	40	40	50	50
Order Amount	100	100	100	90	90	100	100	90	100	100
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	50	40	50	30	40	30	40	50	50	40
Order Amount	90	100	100	110	110	100	100	110	90	110
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	50	40	60	40	50	40	30	50	40	50
Order Amount	100	110	100	100	90	110	130	90	120	110
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	40	50	40	50	40	30	40	40	40	40
Order Amount	90	90	110	90	110	100	110	110	110	110
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	40	40	40	50	40	50	40	50	40	40
Order Amount	110	100	90	100	110	100	110	90	100	120
Total Costs	−85.815874									



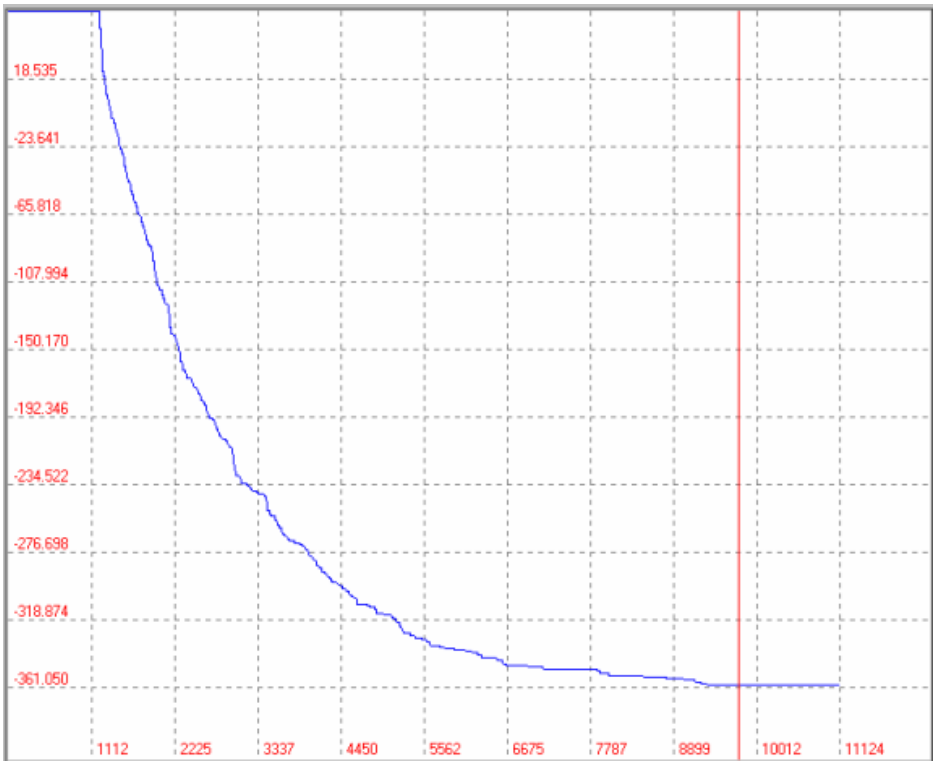
Optimization: Hybrid Parallel (N(100;10))

Solution number	9011									
Number of TU	21									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	-10	0	20	0	10	-10	0	-10	10	0
Order Amount	180	170	60	60	100	150	60	100	100	60
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	0	40	0	0	100	0	0	-10	-20	-20
Order Amount	110	80	60	60	60	140	170	100	100	200
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	60	10	-30	20	0	-10	60	0	0	10
Order Amount	70	160	100	120	160	220	120	60	120	190
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	-50	30	10	0	80	110	-10	-10	0	0
Order Amount	100	170	130	60	60	130	110	200	60	100
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	0	0	10	0	80	0	0	0	110	0
Order Amount	190	60	60	100	80	60	150	60	100	130
Total Costs	−260.662655									



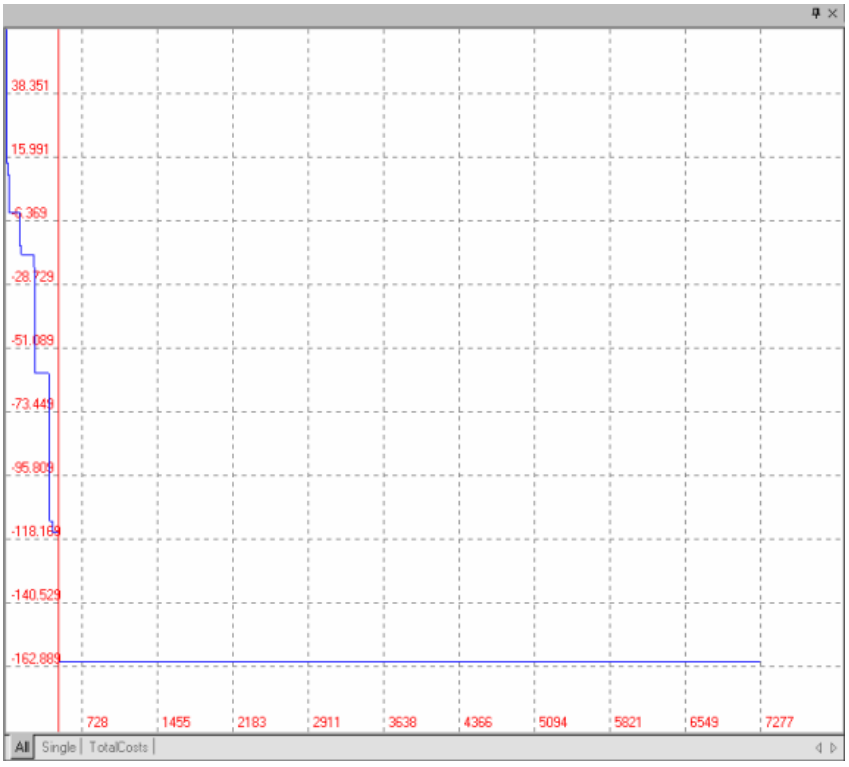
Optimization: Hybrid Seriell (N(100;10))

Solution number	9656									
Number of TU	26									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	0	0	0	0	0	0	0	0	0	0
Order Amount	60	60	60	60	60	60	60	60	60	60
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	0	0	0	0	0	0	0	0	0	-10
Order Amount	60	60	60	60	60	60	60	60	60	100
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	0	0	0	0	0	-10	0	0	0	0
Order Amount	60	60	60	130	60	100	60	60	60	60
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	0	-20	0	0	0	0	0	0	0	0
Order Amount	60	100	60	60	60	60	130	60	60	60
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	0	0	0	0	0	0	0	0	0	-10
Order Amount	60	60	60	60	60	60	60	60	60	100
Total Costs	−359.752507									



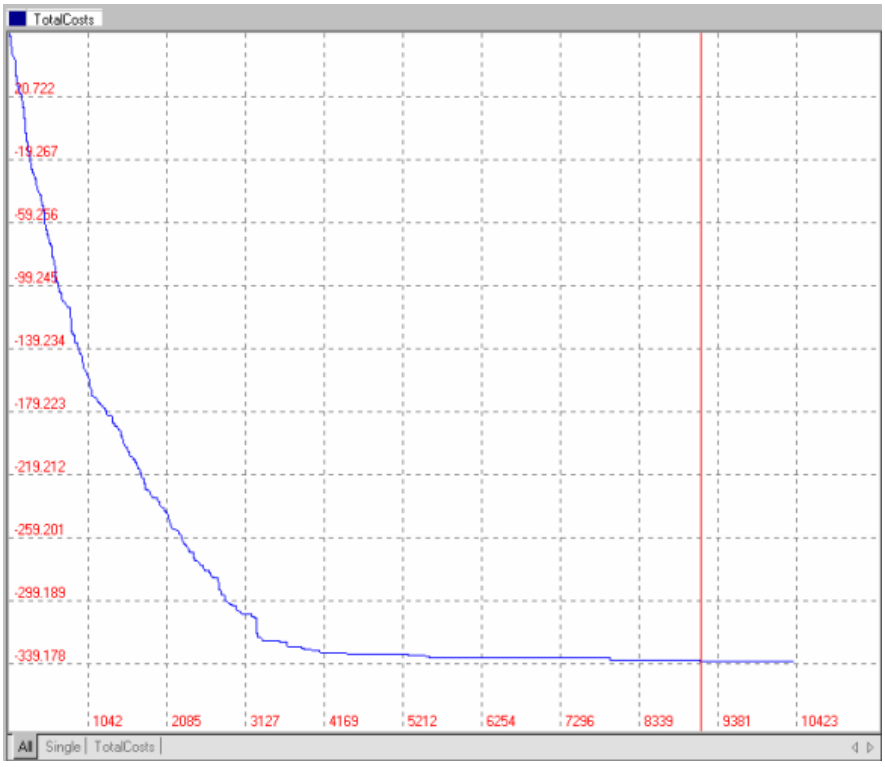
Optimization: GA Generation (N(100;10))

Solution number	498									
Number of TU	18									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	10	10	30	50	60	50	50	20	0	60
Order Amount	130	160	160	170	110	90	100	130	180	120
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	-10	10	90	20	0	10	40	-20	90	110
Order Amount	120	180	160	160	200	190	130	110	150	150
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	40	50	50	40	40	110	120	150	120	90
Order Amount	160	180	130	140	130	110	120	150	120	90
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	20	0	70	70	60	-30	10	30	70	30
Order Amount	140	180	130	130	160	130	170	180	180	190
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	60	70	80	50	30	30	70	100	90	90
Order Amount	190	120	110	150	150	130	120	100	100	140
Total Costs	−161.398704									



Optimization: Tabu Search (N(100;10))

Solution number	9083									
Number of TU	23									
Spoke	1	2	3	4	5	6	7	8	9	10
Order Level	0	0	0	0	0	0	0	0	0	0
Order Amount	60	60	60	60	60	60	60	60	100	60
Spoke	11	12	13	14	15	16	17	18	19	20
Order Level	-10	100	0	0	0	0	0	0	10	0
Order Amount	100	60	60	60	60	60	60	170	160	150
Spoke	21	22	23	24	25	26	27	28	29	30
Order Level	0	0	0	0	0	0	0	0	0	0
Order Amount	60	130	60	60	200	60	60	60	110	60
Spoke	31	32	33	34	35	36	37	38	39	40
Order Level	0	0	0	0	0	0	0	0	0	0
Order Amount	60	60	60	60	130	130	130	130	60	60
Spoke	41	42	43	44	45	46	47	48	49	50
Order Level	-20	0	10	0	0	0	0	10	10	0
Order Amount	200	120	130	60	60	130	60	120	120	60
Total Costs	−337.947927									



References

[Ass80] A. Assad. Modelling of rail networks: Toward a routing / makeup model. *Transportation Research-B*, 14B:101–114, 1980.

[Ayk95] T. Aykin. Networking policies for hub and spoke system with application to the airtransportationsystem. *Transportation Science*, 29:201–221, 1995.

[Cam96] J. Campbell. Hub location and the p-hub median problem. *Operational Research*, 44:923–935, 1996.

[CR86] T. Crainic and J. Rousseau. Multicommodity, multimode freight transportation: A general modeling andalgorithmicframework for the servise network design problem. *Transportation Research-B*, 20B:225–242, 1986.

[DH97] Y. Du and R. Hall. Fleet sizing and empty equipment redistribution for center-terminal transportationnetworks. *Management Science*, 43:145–157, 1997.

[DP77] M. E. Dyer and L. G. Proll. On the validity of marginal analysis for allocating servers in m/m/c queues. *Management Science*, 23(9):1019–1022, 1977.

[Fox66] B. Fox. Discrete optimization via marginal analysis. *Management Science*, 13(3):210–216, 1966.

[Fu94] M. C. Fu. Optimization via simulation: a review. *Ann. of Oper.Res.*, 42:199–247, 1994.

[JSY83] P. Jaillet, G. Song, and G. Yu. Airline network design and hub location problems. *Location Science*, 4:195–212, 1983.

[Käm04] Michael Kämpf. Projektseite > caos > hauptseite. Internet http://www-user.tu-chemnitz.de/ mkae/de/Projects/CAOS/index.php, Stand 12:44, 27. Sep 2006, November 2004.

[Käm05] Michael Kämpf. Softwaresystem zur modellierung, simulation und optimierung ökonomischer problemstellungen. Internet http://www-user.tu-chemnitz.de/ mkae/Publications/WorkingPapers/CSR/CSR.pdf, Stand 12:44, 27. Sep 2006, April 2005.

[Kea92] M. Keaton. Designing railroad operating plans: A dual adjustment method for implementinglagrangianrelaxation. *Transportation Science*, 26:263–279, 1992.

[KKN03] P. Köchel, S. Kunze, and U. Nieländer. Optimal control of a distributed service system with moving resources: Applicationtothe fleet sizing and allocation problem. *International Journal of Production Economics*, 81–82:443–459, 2003.

[KL76] E. Koenigsberg and R. C. Lam. Cyclic queue models of fleet operations. *Oper. Res.*, 24:516–529, 1976.

[Köc05] P. Köchel. Some considerations on optimisation in logistic systems. In *Proceedings of the 8th Intern. Symposium on Operational Research SOR'05*, pages 219–226, Nova Gorica, Slovenia, September 2005.

[MW84] T. Magnanti and R. Wong. Network design and transportation planning: Model and algorithms. *Transportation Science*, 26:1–55, 1984.

[NY00] Alexandra M. Newman and Candace Arai Yano. centralized and decentralized train scheduling for intermodal operations. *IIE Transactions*, 32:743–754, 2000.

[OB98] M. O'Kelly and D. Bryan. Hub location with flow economis of scale. *Transportation Research-B*, 32:605–616, 1998.

[OSKSK95] M. O'Kelly, D. Skorin-Kapov, and J. Skorin-Kapov. Lower bounds for the hub location problem. *Management Science*, 41:713–721, 1995.

[Tri82] Kishor Shridharbhai Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. Prentice-Hall, 1st edition, 1982.

[Web80] R. R. Weber. On the marginal benefit of adding servers to g/gi/m queues. *Management Science*, 26(9):946–951, 1980.

[WHW99] P. Wu, J. C. Hartman, and G. R. Wilson. Fleet sizing in the truck rental. Working Report 98W-009, Lehigh University, 1999.

Chemnitzer Informatik-Berichte

In der Reihe der Chemnitzer Informatik-Berichte sind folgende Berichte erschienen:

CSR-02-01 Andrea Sieber, Werner Dilger, Theorie und Praxis des Software Engineering, Oktober 2001, Chemnitz

CSR-02-02 Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski, Energy-Efficient Size Approximation of Radio Networks with no Collision Detection, Februar 2002, Chemnitz

CSR-02-03 Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski, Efficient Algorithms for Leader Election in Radio Networks, Februar 2002, Chemnitz

CSR-02-04 Tomasz Jurdzinski, Miroslaw Kutylowski, Jan Zatopianski, Weak Communication in Radio Networks, Februar 2002, Chemnitz

CSR-03-01 Amin Coja-Oghlan, Andreas Goerdt, André Lanka, Frank Schädlich, Certifying Unsatisfiability of Random $2k$-SAT Formulas using Approximation Techniques, Februar 2003, Chemnitz

CSR-03-02 M. Randrianarivony, G. Brunnett, Well behaved mesh generation for parameterized surfaces from IGES files, M"arz 2003, Chemnitz

CSR-03-03 Optimizing MPI Collective Communication by Orthogonal Structures, Matthias Kühnemann, Thomas Rauber, Gudula Rünger, September 2003, Chemnitz

CSR-03-04 Daniel Balkanski, Mario Trams, Wolfgang Rehm, Heterogeneous Computing With MPICH/Madeleine and PACX MPI: a Critical Comparison, Dezember 2003, Chemnitz

CSR-03-05 Frank Mietke, Rene Grabner, Torsten Mehlan, Optimization of Message Passing Libraries - Two Examples, Dezember 2003, Chemnitz

CSR-04-01 Karsten Hilbert, Guido Brunnett, A Hybrid LOD Based Rendering Approach for Dynamic Scenes, Januar 2004, Chemnitz

CSR-04-02 Petr Kroha, Ricardo Baeza-Yates, Classification of Stock Exchange News, November 2004, Chemnitz

CSR-04-03 Torsten Hoefler, Torsten Mehlan, Frank Mietke, Wolfgang Rehm, A Survey of Barrier Algorithms for Coarse Grained Supercomputers, Dezember 2004, Chemnitz

CSR-04-04 Torsten Hoefler, Wolfgang Rehm, A Meta Analysis of Gigabit Ethernet over Copper Solutions for Cluster-Networking, Dezember 2004, Chemnitz

CSR-04-05 Christian Siebert, Wolfgang Rehm, One-sided Mutual Exclusion A new Approach to Mutual Exclusion Primitives, Dezember 2004, Chemnitz

Chemnitzer Informatik-Berichte

CSR-05-01 Daniel Beer, Steffen Höhne, Gudula Rünger, Michael Voigt, Software- und Kriterienkatalog zu RAfEG - Referenzarchitektur für E-Government, Januar 2005, Chemnitz

CSR-05-02 David Brunner, Guido Brunnett, An Extended Concept of Voxel Neighborhoods for Correct Thinning in Mesh Segmentation, M"arz 2005, Chemnitz

CSR-05-03 Wolfgang Rehm (Ed.), Kommunikation in Clusterrechnern und Clusterverbundsystemen, Tagungsband zum 1. Workshop, Dezember 2005, Chemnitz

CSR-05-04 Andreas Goerdt, Higher type recursive program schemes and the nested pushdown automaton, Dezember 2005, Chemnitz

CSR-05-05 Amin Coja-Oghlan, Andreas Goerdt, André Lanka, Spectral Partitioning of Random Graphs with Given Expected Degrees, Dezember 2005, Chemnitz

CSR-06-01 Wassil Dimitrow, Mathias Sporer, Wolfram Hardt, UML basierte Zeitmodellierung für eingebettete Echtzeitsysteme, Februar 2006, Chemnitz

CSR-06-02 Mario Lorenz, Guido Brunnett, Optimized Visualization for Tiled Displays, März 2006, Chemnitz

CSR-06-03 D. Beer, S. Höhne, R. Kunis, G. Rünger, M. Voigt, RAfEG - Eine Open Source basierte Architektur f"ur die Abarbeitung von Verwaltungsprozessen im E-Government, April 2006, Chemnitz

CSR-06-04 Michael Kämpf, Probleme der Tourenbildung, Mai 2006, Chemnitz

CSR-06-06 Torsten Hoefler, Mirko Reinhardt, Torsten Mehlan, Frank Mietke, Wolfgang Rehm, Low Overhead Ethernet Communication for Open MPI on Linux Clusters, Juli 2006, Chemnitz

CSR-06-07 Karsten Hilbert, Guido Brunnett, A Texture-Based Appearance Preserving Level of Detail Algorithm for Real-time Rendering of High Quality Images, August 2006, Chemnitz

CSR-06-08 David Brunner, Guido Brunnett, Robin Strand, A High-Perpormance Parallel Thinning Approach Using a Non-Cubic Grid Structure, September 2006, Chemnitz

CSR-06-09 El-Ashry, Peter Köchel, Sebastian Schüler, On Models and Solutions for the Allocation of Transportation Resources in Hub-and-Spoke Systems, September 2006, Chemnitz