A Hybrid Method for Solving Large-Scale Supply Chain Problems

Steffen Wolf and Peter Merz*

Distributed Algorithms Group
University of Kaiserslautern, Germany
{wolf,pmerz}@informatik.uni-kl.de

Abstract. The strategic supply chain design problem which allows capacity shifts and budget limitations can be formulated as a linear program. Since facilities are allowed to be opened or shut down during the planning horizon, this problem is in fact a mixed integer problem. Choosing the optimal set of facilities to serve the customer demands is an NP-hard combinatorial optimization problem. We present a hybrid method combining an evolutionary algorithm and LP based solvers for solving large-scale supply chain problems, which takes its power from filtering out infeasible solutions. The EA incorporating these filters is shown to be faster than the MIP solver ILOG CPLEX in most of the considered instances. For the remaining instances it finds feasible solutions much faster than the MIP solver.

1 Introduction

Consider the following optimization problem: Given a set of customers to serve, a set of products to manufacture and deliver, projections for the demand of the customers as well as for transportation and purchasing costs, the aim is to find optimal combinations of facilities to deliver the goods for a predefined number of time periods. This is a typical problem in supply chain design.

Many simple facility location problems have been studied in the past [1–3]. However, in real world problems the number of locations that can be opened or shut down is often implicitly limited by budget constraints. Also, in real world problems capacity limitations are usually not fixed, but can be extended.

In this work, we use a more advanced supply chain model, that introduces more freedom of choice, but also increases the complexity and therefore the optimization effort. We propose a hybrid method comprising filters and an evolutionary algorithm to approach large scale supply chain design problems. We show in experiments on various problems of different sizes how much can be gained using the filters and compare the results to ILOG CPLEX [4].

The paper is organized as follows. In Section 2, the mathematical formulation for the considered supply chain optimization problem is given. Methods for filtering out infeasible solutions and thereby reducing the search space are presented

* This work was partially supported by the Rhineland-Palatinate Cluster of Excellence 'Dependable Adaptive Systems and Mathematical Modelling'.

in Section 3. In Section 4, we present a simple evolutionary algorithm incorporating these filters and give results in Section 5. Conclusions and an outline for future research are provided in Section 6.

2 Supply Chain Optimization

Many supply chain optimization problems can be formulated as a Linear Program (LP) or a Mixed Integer Linear Program (MIP). This allows standard algorithms and also standard software to be used, e. g. CPLEX. In this work, we use the mathematical model given in [5], which is formulated as an MIP.

2.1 Problem Definition

The model uses a number of parameters, which can be best described in form of matrices. In the following, $X^t_{l,p}$ denotes the value for parameter or variable X for time period t, location l and product type p, and $X^t_{l,l',p}$ denotes the value for the arc from facility l to l'. If X does not depend on the location, the product type or the time period, the corresponding index l, p or t is not written.

Parameter $D^t_{l,p}$ contains the customer demand, $PC^t_{l,p}$ the unit purchase cost, $TC^t_{l,l',p}$ the transportation cost, $IC^t_{l,p}$ the inventory carrying cost and OC^t_l the operation cost. The shutdown cost SC^t_l, the facility setup cost FC^t_l and the cost for shifting capacity from one facility to another facility $MC^t_{l,l'}$ are limited by a budget B^t. Non-invested capital ξ^t can be saved with interest rate β^t. The capacity of a facility is limited by an upper bound $\overline{K}^t_l$ and a lower bound $\underline{K}^t_l$, the latter also denotes a minimal throughput for this facility. A unit capacity consumption factor $\alpha_{l,p}$ allows to let different product types or facilities have different influences on the capacities.

The variables in this model are the binary operational status δ^t_l of the facilities, where $\delta^t_l = 1$ denotes that facility l is operating, and $\delta^t_l = 0$ that it is not operating in time period t. Closed facilities have no capacity, and once a facility falls below the minimum capacity $\underline{K}^t_l$ it has to be closed. The other variables are the product flow $x^t_{l,l',p}$, the amount of purchased products $b^t_{l,p}$, the amount of products to be stored for the next time period $y^t_{l,p}$, and the amount of capacity to be shifted $z^t_{l,l'}$. Some values for the first time period are pre-defined, such as the amount of products held on stock, the non-invested capital and the operational status. All parameters have positive values, only the shutdown costs SC^t_l can be negative, e. g. when governmental subsidies are given.

The cost function C to be minimized is:

$$C = \sum_{\substack{t\in T\\ l\in L\\ p\in P}} PC^t_{l,p} b^t_{l,p} + \sum_{\substack{t\in T\\ l,l'\in L, l\neq l'\\ p\in P}} TC^t_{l,l',p} x^t_{l,l',p} + \sum_{\substack{t\in T\\ l\in L\\ p\in P}} IC^t_{l,p} y^t_{l,p} + \sum_{\substack{t\in T\\ l\in L}} OC^t_l \delta^t_l \quad (1)$$

Here, $T = \{1, \ldots, n\}$ is the set of time periods, P the set of product types and L the set of all facilities, e. g. plants, distribution centers and customers.

There are three pre-defined sets of facilities: $S^{\mathrm{o}} \subseteq L$ the facilities to be opened, $S^{\mathrm{c}} \subseteq L$ the facilities to be closed, and $L \setminus (S^{\mathrm{o}} \cup S^{\mathrm{c}})$ the non-selectable facilities. The following equations give the current capacity K_l^t for all facilities:

$$\forall l \in S^{\mathrm{c}}, t \in T: \; K_l^t = \overline{K}_l^1 - \sum_{\substack{\tau \in \{1,\dots,t\} \\ i \in S^{\mathrm{o}}}} z_{l,i}^{\tau} \tag{2}$$

$$\forall l \in S^{\mathrm{o}}, t \in T: \; K_l^t = \sum_{\substack{\tau \in \{1,\dots,t\} \\ i \in S^{\mathrm{c}}}} z_{i,l}^{\tau} \tag{3}$$

$$\forall l \in L \setminus (S^{\mathrm{o}} \cup S^{\mathrm{c}}), t \in T: \; K_l^t = \overline{K}_l^t \tag{4}$$

The capacity for facilities in S^{c} is defined by the parameter $\overline{K}_l^1$ and can only be reduced by shifting capacity to another facility (2). New facilities start with no capacity and have to receive capacity shifts before they can operate (3). The capacities of the remaining facilities is given in the parameter $\overline{K}_l^t$ and cannot be changed (4). Using these equations the remaining constraints can be formulated as:

$$\forall l \in L, p \in P, t \in T: b_{l,p}^t + \sum_{l' \in L \setminus \{l\}} x_{l',l,p}^t + y_{l,p}^{t-1} = D_{l,p}^t + \sum_{l' \in L \setminus \{l\}} x_{l,l',p}^t + y_{l,p}^t \tag{5}$$

$$\forall l \in L, t \in T: \underline{K}_l^t \delta_l^t \leq \sum_{p \in P} \alpha_{l,p} \left(b_{l,p}^t + \sum_{l' \in L \setminus \{l\}} x_{l',l,p}^t + y_{l,p}^{t-1} \right) \leq K_l^t \leq \overline{K}_l^t \delta_l^t \tag{6}$$

$$\forall l \in S^{\mathrm{c}}, t \in T: K_l^t \geq \delta_l^t \epsilon \tag{7}$$

$$\forall t \in T: \sum_{\substack{l \in S^{\mathrm{c}} \\ l' \in S^{\mathrm{o}}}} MC_{l,l'}^t z_{l,l'}^t + \sum_{l \in S^{\mathrm{c}}} SC_l^t \left(\delta_l^{t-1} - \delta_l^t \right) + \sum_{l \in S^{\mathrm{o}}} FC_l^t \left(\delta_l^{t+1} - \delta_l^t \right) + \xi^t = B^t + (1 + \beta^{t-1}) \cdot \xi^{t-1} \tag{8}$$

$$\forall t \in T, l \in S^{\mathrm{c}}: \; \delta_l^t \geq \delta_l^{t+1} \tag{9}$$

$$\forall t \in T, l \in S^{\mathrm{o}}: \; \delta_l^t \leq \delta_l^{t+1} \tag{10}$$

In short, constraints (5) state that all demands have to be fulfilled either by purchasing products, delivery from another location or by having them stored in the previous time period. Constraints (6) ensure that a facility cannot produce, receive or store more than its capacity in each time period. Moreover, its maximum and minimum allowed capacity cannot be exceeded. Constraints (7) (with $\epsilon > 0$ sufficiently small) ensure that a facility has to be closed when all its capacity has been removed. Constraints (8) give the budget limitations, which include shutdown and setup costs as well as moving capacity costs, and allow unused budget to be saved (with interest) for a later time period. Finally, constraints (9) and (10) state that facilities that have been closed cannot be re-opened, and facilities that have been opened cannot be closed again. Obvious constraints, such that all values have to be real and positive, have been left out.

The problem can be reduced to a static uncapacitated facility location problem which was shown to be NP-hard in Cornuéjols *et al.* [6].

2.2 Related Work

In [5] various small problems for this model and a slightly improved version were solved to optimality by off the shelf commercial software. Velásquez and Melo [7] have introduced variable neighborhood search heuristics to approach larger problems. Here, a linear programming (LP) solver was used to solve the LP subproblem that is obtained by fixing all δ_l^t to values determined by the heuristics. Based on the total cost, the heuristics would then slightly change the δ-values and calculate the new δ-combination. However, no explicit effort has been made to filter out infeasible combinations. So, the LP solver often takes a non-negligible time to report the infeasibility of the considered combination. Since an infeasible combination is undesired, this though small calculation time can be considered as wasted.

3 Search Space Reduction

We propose a way of improving the heuristics in [7]. Again we separate the generation of δ-combinations from the LP calculations. The generation of combinations can be seen as a combinatorial optimization problem with a very complex cost function. Since most computation time is used in the LP calculations we seek to avoid this calculation as often as possible.

Our method of choice is filtering. Once a δ-combination has been chosen, a number of simplified constraint checks are applied. Only combinations passing these checks will be presented to the LP solver. This way, we can avoid many unnecessary calls to the LP solver, allowing it to reuse information from previous runs without the interruption caused by an infeasible combination. Also, by using simplified and explicit constraints, these checks can be carried out faster.

3.1 Shutdown and Setup Costs

A first check that can be applied is to calculate the sum of shutdown and setup costs and check whether the amount stays within the budget limits:

$$\forall t \in T : \sum_{l \in S^c} SC_l^t \left(\delta_l^{t-1} - \delta_l^t\right) + \sum_{l \in S^o} FC_l^t \left(\delta_l^{t+1} - \delta_l^t\right) \leq B^t + (1 + \beta^{t-1}) \cdot \hat{\xi}^{t-1}$$

Here, $\hat{\xi}$ is the remaining capital when capacity shift costs are ignored. This inequality is a relaxation of (8). The check filters out all combinations where too many changes are scheduled. In real world examples only a very small number of facilities are supposed to be opened, but a larger number of possible locations may be provided to choose from. Once a combination is chosen this check can be applied iteratively for each time period. When the accumulated costs exceed the budget (plus accumulated interests) the combination can be discarded, aborting all following checks. Also, a specific facility can be identified whose opening or shutdown caused the check to fail.

3.2 Capacity Shifts

In the model, capacity can only be shifted from existing to new facilities. Also, all capacity from a facility that is to be closed has to be shifted to new facilities. Since a maximum capacity is given, this may not always be possible. A quick check can reveal whether there are enough new facilities to receive the capacity of the existing facilities. Again, this has to be verified for each time period.

This analysis also gives an amount of capacity $\hat{z}$ which has to be shifted, thus creating capacity shift costs that have to be covered by the budget. Since these costs depend on time period and involved facilities, we cannot determine the exact cost, but only give lower bounds. A first lower bound takes the minimum of all entries in MC. A better lower bound takes account of the facilities that need to lose or receive capacity and uses the minimum of only those entries in MC. A more sophisticated lower bound may also include the time period. However, capacity shifts can be scheduled in any time before closing the facility, so this does not not yield much. Denoting this minimum as $MC_{\min}$, these checks can be formulated as:

$$MC_{\min} \cdot \hat{z} + \sum_{l \in S^c} SC_l^t \left(\delta_l^{t-1} - \delta_l^t\right) + \sum_{l \in S^o} FC_l^t \left(\delta_l^{t+1} - \delta_l^t\right) \leq B^t + (1 + \beta^{t-1}) \cdot \hat{\xi}^{t-1}$$

3.3 Customer Demands

The parameter D gives the customer demands. All demands have to be fulfilled. However, customers may buy the needed products from an external supplier. This is accounted for by introducing some form of penalty costs. In order for the customer demands to be fulfilled there has to be an operating facility and a path from this facility to the customer. If this path passes other facilities (e. g. plants – warehouses – customers), they also have to be operating. Calculating all paths for all customers may be expensive, considering that the path may reach over multiple time periods since products can be stored. We therefore determine for each customer and product type only the set of facilities that can deliver the goods directly. Out of this set at least one facility has to be operating.

4 Evolutionary Algorithm for the Supply Chain Problem

In this section we present a simple Evolutionary Algorithm (EA) that incorporates these checks. The algorithm is shown in Fig. 1. In this simple EA we use only mutation. The population size is fixed at μ. In each generation a mutation operation is used to generate λ valid children. After the mutation operator is applied the resulting combination is presented to the filter. If one of the checks fails, the combination is removed and another offspring is created. Only those combinations that pass all checks are given to the LP solver. The combination is also discarded, if the LP solver states infeasibility. Once λ valid offspring combinations are found and evaluated, the combinations to form the next generation are selected. Out of the μ original and the λ offspring combinations, the μ best

```
function EA( κ, λ, μ: Integer );
begin
  for i := 0 to μ−1 do p[i] := Init ();
  for iter := 0 to κ−1 do
  begin
    for i := 0 to λ−1 do
    begin
      repeat
        p[μ+i] := Mutate( p[i mod μ] );
        if CheckFilters(p[μ+i]) then LPsolver(p[μ+i]);
      until isFeasible (p[μ+i]);
    end;
    p := Select (p);
  end;
  return Best(p);
end;
```

Fig. 1. The Evolutionary Algorithm Framework

individuals are chosen, following a $(\mu + \lambda)$ selection paradigm. This procedure is repeated κ times.

To build the initial generation, random combinations can be used as well as combinations based on the LP-relaxation solution of the supply chain problem. In the random approach, a combination is created by scheduling an opening/ closing of a facility with probability p, choosing the time period at random. When no information about the maximum number of possible shutdowns and openings is available, this procedure has to be repeated for some possible values of p. Once a feasible combination is found, p can be adjusted to match the best known combination. The LP-relaxation approach works by solving the LP problem which results when dropping the constraint that all δ have to be binary. The solution found may still assign binary values to some δ. The remaining non-binary values can be used to 'guess' the values of other δ. A binary δ-combination can be generated from this knowledge by random assignment of 0 or 1, using the δ-values as a probability. However, these combinations tend to be infeasible since they often schedule more openings/shutdowns than the budget allows. Reducing or limiting the number of opening/shutdowns helps to find feasible combinations.

Instead of storing all δ-values, we use a more compact encoding. For each selectable facility l we store the time period γ_l of its opening/shutdown. Thus, each individuum can be described by only $s = \mid S^{\mathrm{o}} \cup S^{\mathrm{c}} \mid$ integer variables instead of $\mid T \mid \cdot \mid L \mid$ binary variables. The operational status δ can easily be extracted from this encoding using the following formula:

$$\delta_l^t = 1 \leftrightarrow (l \in S^{\mathrm{o}} \wedge t \geq \gamma_l) \vee (l \in S^{\mathrm{c}} \wedge t < \gamma_l) \vee l \in L \setminus (S^{\mathrm{o}} \cup S^{\mathrm{c}}) \tag{11}$$

The mutation operator used in this EA changes the time period for the opening/shutdown of a random facility to a random value $\gamma_l \in [2, n+1]$. To schedule

an opening/shutdown for period $\gamma_l = n+1$ denotes that the facility should not be opened/closed during the planning horizon. We have also tried other mutation operators, but the multiple consecutive application of the described operator gave the best results.

5 Experiments

To show the effects of these filters we ran experiments on various supply chain instances. Since standard MIP software like ILOG CPLEX [4] can solve smaller problems in very short time, we concentrated on larger instances. Finding real world data for those larger instances proved to be difficult, so we used randomly generated instances as described in [8]. Nine instances were generated with up to 70 selectable and 55 non-selectable locations, 5 product types and 8 time periods. In all instances there are two kinds of distribution centers (DCs). All products have to be transported from the plants to central DCs, and from there to regional DCs before they arrive at the customer sites. Each DC can store products for later time periods. The DCs are the selectable facilities in these instances, all other facilities (such as customers and plants) are non-selectable. The capacities of the locations are limited, but those limits can be changed during the planning horizon according to the model.

In a first set of experiments we were interested in the number of δ-combinations that remain when using the filters. Table 1 shows the results for six small instances (H1, ..., H6). Since this analysis is very expensive – it requires calculating or checking all δ-combinations – it was not applied to the larger instances. The first line gives the number of possibilities to schedule openings/shutdowns for $s = \mid S^{\mathrm{o}} \cup S^{\mathrm{c}} \mid$ locations in $n = \mid T \mid$ time periods: n^s. Most of these combinations are infeasible due to the budget constraints. After the first check the number of combinations is already drastically reduced. Another large amount of combinations is filtered out by the capacity shift checks. The remaining number is shown in the third line. However, as stated in Section 3, the infeasibility of some combinations cannot be recognized by these simple checks. The number of feasible combinations is shown in the fourth line. This number was determined by handing all remaining combinations to the LP solver. The last line shows the influence of the demand satisfiability check. This check filters out combinations with high penalty costs regardless of their feasibility, so the remaining number of combinations is often smaller than the actual number of feasible combinations. Only in problem H3 no combination lead to unfulfilled demands.

The effect of unfulfilled demands can be seen in Fig. 2. The figure shows the calculated costs for some random combinations for problems H1 and N7. In H1 many combinations can be found with costs around $C \approx 10^6$, near the optimum (953 824), but there are some "bands" of combinations with higher costs. All these combinations have unfulfilled demands, because some required facility was closed or not opened, which generates penalty costs. In problem N7 the bands are not that clear, but they also exist here. Problem N7 shows even more bands than in H1. In fact, every dot in this plot above $0.35 \cdot 10^7$ belongs

Problem	H1	H2	H3	H4	H5	H6
Total	3^{30}	4^{30}	3^{30}	3^{45}	3^{30}	3^{30}
Shutdown/Opening Cost Check	278 053	8 881 793	101 626	1 395 442	106 245	158 415
Capacity Shift Check	33 938	517 723	11 747	161 352	9 410	11 890
Feasible Combinations	25 539	322 352	8 321	111 009	6 795	7 671
Demand Satisfiability Check	21 052	299 508	11 747	134 802	3 997	8 072

Table 1. Number of δ-combinations remaining after the filters

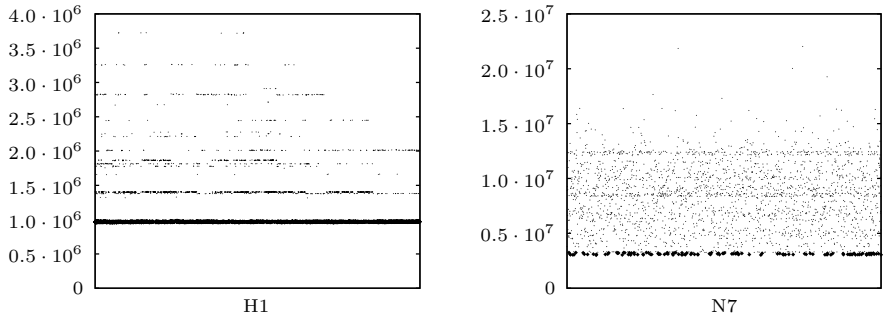

Fig. 2. The distribution of the objective function values for problems H1 and N7

to a higher band. We have discovered these bands in all considered problems, with the exception of H3. An explanation for these bands is the possibility to close (or not open) required facilities. The longer a required facility is closed the higher is the penalty cost. The penalty cost is supposed to be much larger than the differences between the feasible solutions. So for each time period and each required facility a different penalty is applied, which creates the observed bands. The demand satisfiability check filters out these expensive combinations, leaving only the lowest band, marked by bold dots in the figure. These remaining combinations are only up to 10 % more expensive than the optimal or – in case of the larger problems – the best known solution.

In the next set of experiments we were interested in the overall performance of our proposed EA. We fixed $\mu = 5$ and $\lambda = 35$, and used CPLEX 10.1 to evaluate the fitness of each δ-combination, i. e. solve the underlying LP. The number of generations was set to $\kappa = 50$, so CPLEX had to evaluate at least $\mu + \lambda \cdot \kappa = 1755$ combinations. These parameters are a compromise between computation time and solution quality. All CPU times reported here refer to a 3 GHz Pentium IV. The results are averaged over 10 runs.

Figure 3 shows the averaged behavior of the EA for two different problems. In each case the first feasible solution is found almost instantly. In problem N7 the first solution already exceeds the best known solution by only 3 %. This best known solution is reached in 4 of the 10 EA runs after 30-40 min, and also by CPLEX after 25 h. The figure also shows the behavior of the EA when the

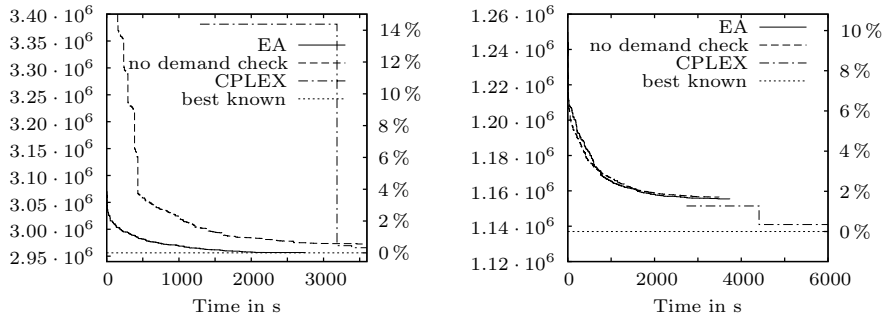

Fig. 3. Behavior of the EA with and without demand check for problems N7 and N19

demand check is not active. Without this check the cost of the first solution is almost four times as high as the best known solution, also the improvement is very slow. To compare the EA to standard software we show the behavior of CPLEX in the same plot. CPLEX in its standard configuration does not find a feasible solution for about 20 min, and the first solution it finds is worse than the first solution of the EA.

In problem N19 the influence of the demand check is negligible. Here, only a small amount of δ-combinations can be found in higher bands, and therefore be cut off. The EA finds a good starting solution and steadily improves it. However, it does not reach the best known solution, but converges between 1.5 % and 1.8 % above it. CPLEX takes about 45 min to find a feasible solution, but this time it is already better than the best solution found by the EA. The best known solution was found by CPLEX after four days.

Problem instance N19 was the worst instance for the EA. Problem N20 showed similar results like N7. We therefore omit the plot. The main difference to N7 is that the computation times are roughly 5 times as high, due to the problem size. However, the best known solution in N20 was found in an EA run after about 4 h, while CPLEX takes about 3 h to find a first feasible solution and does not find the best known solution in the first 30 h.

Table 2 shows the average number of combinations that were filtered out by the proposed filters in an EA run. These numbers are significantly lower than in Table 1, but the filters are still useful, since they filter out about 100 times more combinations than feasible ones, leaving only a small percentage of unnecessary calculations. In our experiments an LP evaluation took about 1-8 s, depending on the problem. An LP run on an infeasible combination took only 0.1-0.8 s, so it is roughly ten times faster. However, CPLEX cannot start from a good LP solution in the next evaluation, so the evaluations after such infeasible runs are slowed down. The advantage of the filters is clear. They can be applied thousand times a second and do not affect the underlying LP solver. Without these filters the EA would show very poor results.

Problem	N7	N19	N20
Shutdown costs too high	191068	193337	148843
Capacity shifts impossible	22108	26967	19450
Demand not met	1622	18	2
Still infeasible	32	16	94
Feasible	1755	1755	1755

Table 2. Average number of δ-combinations filtered out by the filters in an EA run

6 Conclusions

We have proposed an EA using filters to solve large-scale supply chain design problems. We have showed that the EA can find feasible solutions very quickly. The filters reduce the search space significantly and allow the EA to even find the optimal solution in some problems. However, in problems that show many combinations in the lower band, the demand filter cannot be applied efficiently, so the EA may not find the optimal solution and is outperformed by CPLEX. Our method still has an advantage in these cases, since it finds feasible solutions and even good solutions long before CPLEX. These can be given to CPLEX as a starting solution, in order to improve the performance of the MIP solver. This is an issue of future research.

References

1. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. John Wiley & Sons, New York (1988)
2. Resende, M.G.C., Werneck, R.F.: A hybrid multistart heuristic for the uncapacitated facility location problem. European Journal of Operational Research **174** (2006) 54–68
3. Zhang, J.: Approximating the two-level facility location problem via a quasi-greedy approach. In: SODA'04: Proceedings of the 15th annual ACM-SIAM symposium on Discrete algorithms, Philadelphia, PA, USA, Society for Industrial and Applied Mathematics (2004) 808–817
4. ILOG S.A.: ILOG CPLEX User's Manual (2006) Gentilli, France. `http://www.cplex.com/`.
5. Melo, M.T., Nickel, S., Saldanha da Gama, F.: Dynamic multi-commodity capacitated facility location: a mathematical modeling framework for strategic supply chain planning. Computers & Operations Research **33** (2006) 181–208
6. Cornuéjols, G.P., Nemhauser, G.L., Wolsey, L.A.: The uncapacitated facility location problem. In Mirchandani, P.B., Francis, R.L., eds.: Discrete Location Theory. Wiley, New York (1990) 119–171
7. Velásquez, R., Melo, M.T.: Solving a large-scale dynamic facility location problem with variable neighbourhood and token ring search. In: Proceedings of the 39th ORSNZ Conference, Auckland, NZ (2004)
8. Melo, M.T., Nickel, S., Saldanha da Gama, F.: Large-scale models for dynamic multi-commodity capacitated facility location. Technical Report 58, Fraunhofer Institut for Industrial Mathematics (ITWM), Kaiserslautern, Germany (2003) Available at `http://www.itwm.fhg.de/`.