A DYNAMIC MULTICOMMODITY NETWORK FLOW PROBLEM FOR LOGISTICS NETWORKS

Sebastian Raggl(a), **Judith Fechter**(b,d), **Andreas Beham**(c,d)

(a)(b)(c) Heuristic and Evolutionary Algorithms Laboratory
School of Informatics, Communication, and Media
University of Applied Sciences Upper Austria,
Softwarepark 11, 4232 Hagenberg, Austria

(d) Institute for Formal Models and Verification
Johannes Kepler University Linz,
Altenberger Straße 69, 4040 Linz, Austria

(a)sebastian.raggl@fh-hagenberg.at, (b)judith.fechter@fh-hagenberg.at, (c)andreas.beham@fh-hagenberg.at

ABSTRACT

We present a dynamic multicommodity minimum cost network flow problem with storage at the nodes and optimal supply for modeling operations within a logistics network.
The model can be used to evaluate critical business decisions such as the amount of required resources for storage transportation as well as optimal supply policies.
A generator for test instances was written in order to evaluate the performance of the different solution strategies. Using a number of differently sized randomly generated problem instances we compare the execution time and the memory demand of two methods for solving the problem. The first is solving the whole problem formulation directly using general purpose linear programming solvers implemented in IBM Ilog CPLEX. In the second approach we attempt to split the model into two parts and link them together in an optimization network. We analyze the quality of the link and propose possibilities to improve the two step approach through input parameter variation

Keywords: Dynamic networks, Minimum cost flow problem, Multicommodity flows, Dynamic Lot Sizing Problem

1. INTRODUCTION

Network flow problems have been used to model a wide variety of systems and therefore network flow literature distinguishes a large number of problems and variants. A good overview over problems as well as algorithms can be found in (Ahuja et al. 1993, Kotnyec 2003).

When modeling a logistics network there will likely be more than one commodity that has to be transported across the network and the transport capacities must be shared between all the commodities. This kind of problem is known in the literature as a multicommodity network flow problem as opposed to the classical single-commodity network flow problem.

Another major distinction is between static and dynamic networks. Dynamic network flows were first introduced by (Ford and Fulkerson 1958). They described the problem of maximum flows with transport time on the arcs. Their solution was to transform the dynamic network into a so called time-expanded network then solve this problem using classical static methods and finally transform the solution back into the original dynamic form. It has been shown that the same approach is also valid for dynamic minimum cost multicommodity flows (Fonoberova 2010). In this case the problem that has to be solved on the time-extended network is a minimum-cost multicommodity network flow problem (Assad 1978, McBride 1998, Castro and Nabona 1996, Karakostas 2008).

Time is an essential factor in logistics networks because of a number of reasons. While the classical static problem only considers the transport capacities and costs on the arcs the dynamic problem can model storage capacity as well as costs at the nodes which are equally important. In order to make meaningful statements about the network the model must take into account that supplies and demands that change over time. Supply policies that are implemented in many companies, e.g. using stock reorder and order-up-to levels govern the time and amount of items that have to be purchased. These policies, if parameterized wrongly, put a large burden on the efficiency of the network. In the presented case, the model considers both the movement as well as the acquisition of commodities.

The network flow problem presented in this paper is a dynamic multicommodity minimum cost network flow problem with storage at the nodes and optimal supply. The model also allows for multiple modes of transports as well as multiple sinks and sources for each commodity.

2. OPTIMIZATION MODEL

The goal of the model is to describe operations in supply chains and optimize the associated costs arising out of the acquisition, storage, handling, and transportation of commodities. In general, a logistic network can be described as a set of transport modes M and a graph $G = (V, E)$ with nodes (V) and edges (E). An edge (u, v, m) with $u, v \in V$ and $m \in M$ describes a link between two nodes making use of a certain transport mode. The notation $E^+(u)$ and $E^-(u)$ is used to describe the set of all the edges going to and coming from a certain node u. The logistic network is used to execute the flow of a set of commodities C. The operations are described as a series of discrete time steps T. In this model all commodities that are to be purchased emerge in a virtual node "q" from which it must be distributed. This option was chosen to let the model decide, taking into account logistic cost and capacities, the delivery node for a certain purchase.

Figure 1 shows a small exemplary network with two large storage nodes and three smaller sink nodes. The supply appears in the virtual source node and because there is no storage capacity in this node it has to be delivered to one of the two warehouses immediately. When there is a demand at a store the network transports the commodities from the warehouses to this store.

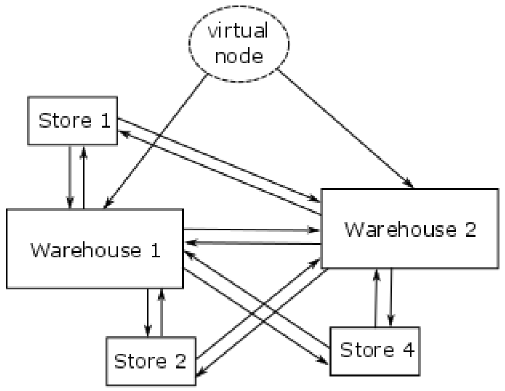

Figure 1 Example network

The number of capacity units stored in node u at any time is a variable called $f_{utc} \in \mathbb{R}^+$. It is constrained by a lower and upper limit namely a_u and b_u. The cost of storing one capacity unit at node u for one time step is called α_u.

Each mode m has a certain maximum capacity z_m. The capacity is to be distributed over all edges that specify that mode. Not all of the capacity needs to be assigned. In fact it will be a goal to minimize the employed capacities. So for each edge and each time step the decision has to be made how much capacity is to be assigned. That is what the continuous variable $x_{ect} \in \mathbb{R}^+$ is used for. The modes are also associated with a certain cost per unit of capacity transported. These transport costs together with the handling costs that occur in the source and target nodes of an edge are called β_e the total cost of transporting one unit of any commodity over edge e.

The amount of each commodity c initially stored in each node u is given by $i_{uc} \in \mathbb{R}^+$. The demand $d_{uct} \in \mathbb{R}^+$ is given for each node, commodity and time step. When to buy how much of which commodity is to be determined by the model and described by the variable $s_{ct} \in \mathbb{R}^+$.

The price of one unit of commodity c is given as γ_c. To be able to account for fixed cost per order there needs to be an additional variable $o_{ct} \in \{0, 1\}$ which indicates whether an order was placed. The value of the fixed costs per order is given by δ_c.

The factor ε_c is used to convert between pieces of a commodity which is the unit used by demand, supply and initial stock and transport units which are used to describe how much is stored or transported

The goal is to minimize the sum of all acquisition, storage, handling, and transportation costs:

$$\sum_c \sum_t \left(\sum_e x_{ect}\, \beta_e + \sum_u f_{uct}\, \alpha_u + s_{ct}\, \gamma_c + o_{ct} \delta_c \right)$$

subject to:

$$f_{uct-1} - d_{uct}\, \varepsilon_c + \sum_{e \in E^+(u)} x_{ect} - \sum_{e \in E^-(u)} x_{ect} = f_{uct} \quad \forall u, c, t \quad (1)$$

$$\sum\nolimits_{e \in E^-(q)} x_{ect} = s_{ct}\, \varepsilon_c \quad \forall c, t \quad (2)$$

$$\sum\nolimits_c \sum\nolimits_{e \in E_m} x_{ect} \leq z_m \quad \forall m, t \quad (3)$$

$$\sum\nolimits_c \sum\nolimits_{e \in E^+(u)} x_{ect} \leq p_u \quad \forall u, t \quad (4)$$

$$a_u \leq \sum\nolimits_c f_{uct} \leq b_u \quad \forall u, c, t \quad (5)$$

$$s_{ct}\;\; \varepsilon_c \leq o_{ct} M \quad \forall c, t \quad (6)$$

$$f_{uc0} = i_{uc}\, \varepsilon_c \quad \forall u, c \quad (7)$$

Constraint (1) guarantees that commodities do not simply disappear from the network by requiring that the storage capacity used at each node and time step for each commodity depends on the previous time step as wells as the inflow, outflow and demand at that time step. The supply is not mentioned in the first constraint instead there is constraint (2) which states that the sum of all the flows out of the virtual node named q in the model must equal the supply for each commodity and time step. This is useful because otherwise the supply variable would need another index for the node where it has to be delivered which would increase the complexity of the model considerably. Additionaly a new constraint equivalent to (4) would be needed to limit the processing rate for the supply.

The constraints (3, 4 and 5) all limit the amount of commodities both stored in as well as transported between nodes. The next constraint (6) guarantees that there is an order whenever the supply is greater than zero by requiring the supply to be smaller than the order times the maximal possible order size. The maximum possible order size M is defined to be the sum of the processing rates of all nodes connected to node q.

The final constraint deals with the initially stored commodities.

3. SPLITTING THE MODEL

As shown in Table 1 the effort required to solve this model rises very rapidly. In order to overcome this problem it is possible to split the model into two models, one model that makes decision on the acquisition of commodities, i.e. a purchasing problem and another model that decides on the movement of commodities, i.e. a network flow model. The purchasing model ignores the network aspects but decides when to buy how much of which commodity.

Figure 2 illustrates the two different approaches. On the left is the two step approach described above and on the right is the combined model described in Section 2.

3.1. Purchasing Model

The network flow model gets the supplies calculated by the other model as input and optimizes the transportation and storage in the network.

The purchasing model has the goal to minimize the storage costs and the purchasing costs:

$$\sum_c \sum_t (f_{ct}\,\alpha + s_{ct}\,\gamma_c + o_{ct}\delta_c)$$

subject to:

$$f_{ct-1} - d_{ct}\,\varepsilon_c + s_{ct}\,\varepsilon_c = f_{ct} \qquad \forall c,t \qquad (1)$$

$$\sum_c s_{ct}\,\varepsilon_c \le M \qquad \forall t \qquad (2)$$

$$a \le \sum_c f_{ct} \le b \qquad \forall t \qquad (3)$$

$$s_{ct}\,\varepsilon_c \le o_{ct}M \qquad \forall c,t \qquad (4)$$

$$f_{c0} = i_c \qquad \forall c \qquad (5)$$

The variables and constants have the same meaning but because the network aspect is ignored the variables that originally contained a value per node are now summed over all nodes. The only constraint that does not have a direct equivalent in the combined model is constraint (2). It limits the supply at each time step to the combined processing capacity of all nodes connected to the virtual source node q. This is not needed in the combined model because constraint (4) includes this.

3.2. Network flow model

The second model is very similar to the combined model but since s_{ct} and o_{ct} are now constant instead of variable the purchasing costs in the objective also become a constant. The objective function that needs to be minimized then looks like this:

$$\sum_c \sum_t \left(\sum_e x_{ect}\,\beta_e + \sum_u f_{uct}\,\alpha_u \right) + C$$

Where C stands for the total purchasing costs as defined by the supply matrix s_{ct} the commodity prices γ_c and the fixed cost per order δ_c. All the constraints of the original model remain unchanged except for constraint (6) which can be removed entirely.

This new model even though it looks very similar is much easier to solve because there are a lot less variables. Additionally are all of the remaining variables continuous which means that the problem is no longer a MILP but is now simply a Linear Program.

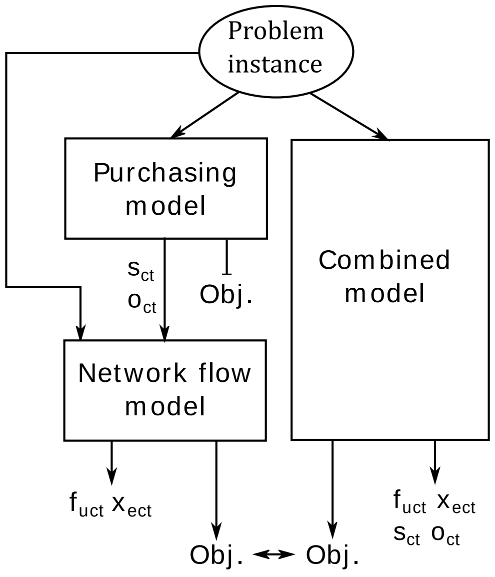

Figure 2 Comparing the two approaches

4. EXPERIMENTS

A generator for random problem instances was written. The problem instances model a retailer with several stores, warehouses, commodities and modes of transport. The warehouses are the sources where commodities are delivered to by the suppliers and the stores are sinks where the commodities sold to the customers disappear from the network. Each problem instance is completely specified by the number of stores, warehouses, commodities days and a seed value for the random number generator. A problem instance is generated using the following steps:

- Store and warehouse nodes are generated and the storage capacities are randomly chosen.
- Transport modes are generated and the nodes are connected by edges.
- Commodities are generated along with prices and conversion factors.

- The nodes are filled with an initial stock of commodities.
- Demands are generated for each store proportional to the stores storage.
- The capacity of the transport modes and the processing capacity of the nodes are chosen so that a feasible solution exists.

4.1. Comparing the two approaches

The first important question is how difficult is it to solve each of the models presented earlier. The models were set up like illustrated in Figure 2 then the time and memory consumption needed to obtain the solution using the two different approaches was recorded for problem instances of varying size.

As a next step the solution quality was compared between the two alternative ways of solving the problem. The idea is that if the solution is good enough it is not even necessary to run the bulky combined model. Finally it was examined if the solution of the two step approach could be improved by varying parameters of the purchasing model used as a first step. If this is the case it should be possible to run an algorithm that solves the problem by repeatedly solving the optimization network and varying these parameters.

5. RESULTS

All three models were implemented and tested using IBM ILOG CPLEX (IBM 2015). All tests were run on a Dell Latitude E6540 with an Intel i7 4810MQ CPU @2.80GHz and 16 GB RAM running Windows 7.

Table 1 Time and memory consumption for differently sized problem instances

C	T	t_{sep}	m_{sep}	t_{com}	m_{net}
10	10	0.41	15.61	0.91	87.61
10	20	0.66	31.09	511.96	290.78
10	30	1.12	28.93	111.29	99.98
100	10	5.02	105.59	404.65	529.21
100	20	38.38	200.98	>12h	
100	30	126.74	281.84	>12h	

Table 1 compares the time and memory needed to find the solution using the two sub models with the integrated model. The column labeled C and T shows the number of commodities and time steps in the problem instance. The size of the logistics network was for all tests set to ten nodes. The values under t_{sep} and t_{com} are the total time in seconds, it took to find the solution using the two separate models and the combined model respectively. Finally m stands for the peak memory consumption in megabytes.

Solving both of the separate models is as expected much faster. For tiny problem instances the runtimes are comparable, but as the problem instances grow larger is the differences are so big that the integrated model does not even find a single solution candidate before the solution of the two models is found. Additionally the solving time of the combined model varies greatly depending on the problem instance.

5.1. Comparing the solution quality

As mentioned before is the two step solution likely to be worse than the one from the combined model. To find out how much worse the objective values of the combined model and the two step approach were compared while varying problem parameters. The problem instances used in this test all have 3 warehouses 7 stores 10 commodities and 30 days and fixed cost per order δ_c was varied between 0 and 100 with step size 10 and the storage cost multiplier α'_u was varied between 0 and 2 with step size 0.2. Each problem instance was solved to optimality with both approaches and the gap between the optimal solution and the approximation is shown in Figure 2. It can be seen that using two separate models can provide solutions that are close to the optimum. In five cases the solution of the optimization network was exactly the same as the one of the combined model. Considering how much faster it is to solve the two models than to solve the integrated model this is quite remarkable.

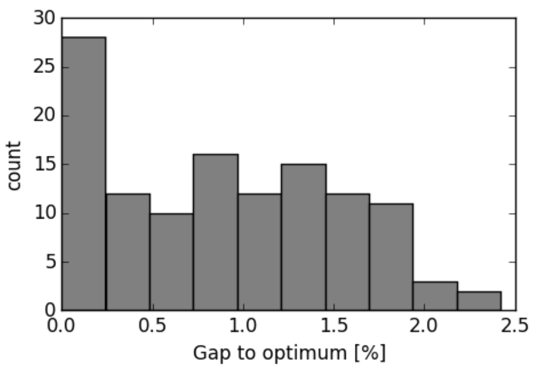

Figure 3 Differences in solution quality between the integrated and the two step model.

5.2. Parameter Variation

For the problem instance with the parameters $\delta_c = 50$ and $\alpha'_u = 1$ the objective value of the network flow model was 1.163% bigger than that of the combined model. The goal of the next test is to determine if a better solution can be found if the parameters of the purchasing model are changed. So the δ_c was again varied between 0 and 100 this time using a step size of 5 and α'_u was varied between 0 and 2 with a step size of 0.1. The purchasing model was parametrized with all the combinations and the resulting supply matrix s_{ct} was passed on to the network flow model. This way we obtain a number of different solutions which can be compared to the solution to the integrated model that was calculated for the previous test.

Figure 3 shows the gap in percent between the optimal and the approximated solution for all the combinations of parameters the purchasing model was set to. While the gap was 1.163% using the original parameters the lowest gap found in this case was 0.119% and was found at $\delta_c = 95$ and $\alpha'_u = 2$. With this parameter grid variation of input parameters we can thus obtain much better results while still having a model that can be solved in reasonable time and close to optimality.

Additionally, Figure 4 shows a rough sketch of the fitness landscape of the optimization problem if we would use a meta-solver to vary these input parameters. The landscape is not as nice to optimize as we would wish as the best solutions can be obtained with only a very small range of parameters that are surrounded by much worse solutions.

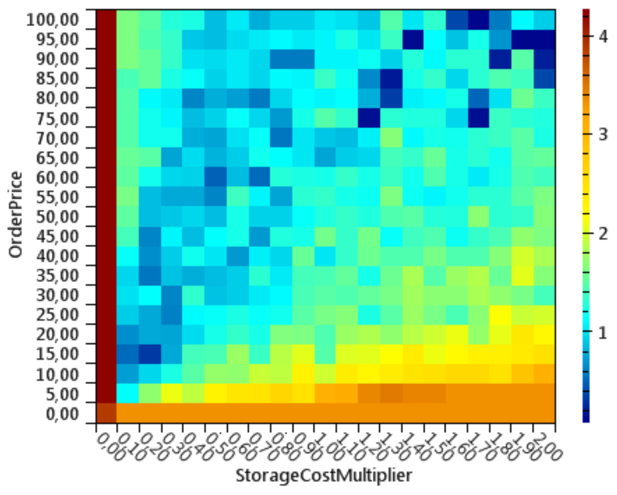

Figure 4 Results of the parameter variation for one instance

6. MODEL APPLICATION

The model describes relevant costs arising due to logistics operations such as handling costs when accepting or commissioning items to or from a warehouse, costs for storing an item for a certain number of time steps and costs that arise out of purchase and transportation. The optimization decides on the transportation and thus determines all movements of commodities in the network. The constraints enforce that demand has to be supplied in time and that commodities that enter the network are being moved should the maximum storage capacity be reached.

One application of the model that is interesting for the real-world is to act as a scenario evaluator for logistics operations. The inputs to this model, i.e. the nodes, edges, when and where items appear and disappear, capacities, cost factors, etc. may be subject to a master level optimization algorithm to which the optimization of this model may appear as an evaluation function. The scenario evaluation aspect is useful in critical business situations such as when new warehouses are to be opened, logistics capacities are expanded or reduced, or also to evaluate environmental effects such as a change in demand. The model can then be used to evaluate the effects on the network and determines costs and capacities.

7. DISCUSSION

Using the two separate models is clearly not equivalent to solving the integrated problem, because in this case the layout of the logistics network does not influence the purchasing decision made by the first model. The network flow model only has to find the best solution for one given supply vector while the combined model optimizes both the flow through the network as well as the supply. So the two step approach is likely to deliver a solution that is suboptimal. What makes the two step solution interesting is that it can be obtained much faster while still providing good solutions. The biggest benefit gets apparent when considering real world problem instances where the number of commodities is likely to be at least an order of magnitude larger and the logistics networks may be bigger. In this case the optimization network can still provide solutions while the combined model becomes utterly unsolvable.

It was shown that by varying the parameters of the purchasing model it is possible to arrive at solutions that are better than the one found with the original parameters. This means that is possible to optimize the combined problem even in cases where the combined MIP is too large to be solved at all. All that is needed is some kind of strategy of how to modify the parameters of the purchasing models. Whatever strategy is used to choose the parameters while optimizing when only the fixed cost per order δ_c and the storage cost multiplier α'_u are used the number of different supply matrices that can be generated is very limited. This can be remedied by adding a dimension to the price array γ_c and turn it into a price matrix γ_{ct}. Using the price matrix the purchasing model can be rewarded for buying a commodity at a certain point in time. It should therefore be possible to generate every valid supply matrix.

8. OUTLOOK

The next step is to implement a meta-solver for the optimization network approach. While changing the parameters randomly might work it is certainly not very efficient. The goal therefore is to find ways to intelligently adept the parameters of the purchasing model using the solutions of the network flow model.

CPLEX offers a way to add a starting point for a MIP problem. In theory it should be possible to combine the two solutions of the sub problems into a starting point for the combined model. This effectively turns the two smaller models into a construction heuristic for the combined model. Using this approach it should be possible to cut down the runtime considerably without having to let go of the advantages of an exact solution.

ACKNOWLEDGMENTS

The work described in this paper was done within the COMET Project Heuristic Optimization in Production and Logistics (HOPL), #843532 funded by the Austrian Research Promotion Agency (FFG).

REFERENCES

Ahuja, R., Magnanti, T. & Orlin, J., 1993. *Network Flows: Theory, Algorithms and Applications.* Upper Saddle River: Prentice-Hall, Inc..

Assad, A. A., 1978. Multicommodity network flows—A survey. *Networks,* 8(1), pp. 37-91.

Castro, J. & Nabona, N., 1996. An implementation of linear and nonlinear multicommodity network flows. *European Journal of Operational Research,* 92(1), pp. 37-53.

Fonoberova, M., 2010. Algorithms for Finding Optimal Flows in Dynamic Networks. In: *Handbook of Power Systems II.* Berlin: Springer, pp. 31-54.

Ford, L. R. & Fulkerson, D. R., 1958. Constructing maximal dynamic flows from static flows. *Operations Research 6,* pp. 419-433.

Ford, L. R. & Fulkerson, D. R., 1962. *Flows in Networks.* s.l.:Princeton University Press.

Hall, A., Hippler, S. & Skutella, M., 2007. Multicommodity flows over time: Efficient algorithms and complexity. *Theoretical Computer Science,* pp. 387-404.

IBM, 2015. *IBM ILOG CPLEX Optimizer.* [Online] Available at: http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/
[Accessed 1 5 2015].

Karakostas, G., 2008 . Faster approximation schemes for fractional multicommodity flow problems. *ACM Transactions on Algorithms,* 4(1), pp. 13:1-13:17.

Klinz, B. & Woeginger, G., 2004. Minimum-cost dynamic flows: The series-parallel case. *Networks,* 43(3), pp. 153-162.

Kotnyec, B., 2003. *An annotated overview of dynamic network flows,* Paris: HAL - Inria.

McBride, R., 1998. Progress Made in Solving the Multicommodity Flow Problem. *SIAM J. on Optimization,* 8(4), pp. 947-955.

AUTHORS BIOGRAPHY

SEBASTIAN RAGGL received his Master of Science in Bioinformatics in 2014 from the University of Applied Sciences Upper Austria, and is a research associate at the Research Center Hagenberg. His research interests include metaheuristic methods applied to combinatorial and simulation-based problems.

JUDITH FECHTER received her Master in Mathematics in 2014 from TU Vienna, Austria, and is a research associate at the Research Center Hagenberg. Her research interests include metaheuristic methods applied to simulation-based optimization problems.

ANDREAS BEHAM received his Master in computer science in 2007 from JKU Linz, Austria, and is a research associate at the Research Center Hagenberg. His research interests include metaheuristic methods applied to combinatorial and simulation-based problems. He is a member of the HeuristicLab architects team.