Alcom-FT Technical Report Series
ALCOMFT-TR-01-13

CNOP - A Package for Constrained Network Optimization*

K. Mehlhorn and M. Ziegelmann**

Max-Planck-Institut für Informatik,
Stuhlsatzenhausweg 85, 66123 Saarbrücken, Germany,
{mehlhorn,mark}@mpi-sb.mpg.de
http://www.mpi-sb.mpg.de/~{mehlhorn,mark}

Abstract. We present a generic package for resource constrained network optimization problems. We illustrate the flexibility and the use of our package by solving four applications: route planning, curve approximation, minimum cost reliability constrained spanning trees and the table layout problem.

1 Introduction

There are a large number of graph and network algorithms that can be efficiently solved in polynomial time, like the shortest path problem, minimum spanning tree problem or flow problems [AMO93]. However, adding a single side constraint involving another cost function to these problems usually makes the problem NP-hard (see Garey and Johnson [GJ79]). Since many practical applications can be modeled using resource constraints, it is of great interest to nevertheless solve the problem efficiently.

In [MZ00] we studied the constrained shortest path problem which is to find a path of minimal cost satisfiying additional resource constraint(s). We extended previous methods of Handler and Zang [HZ80] and Beasley and Christofides [BC89] that solve the problem by first solving a Lagrangean relaxation and then closing the gap by path ranking. In our experiments we found that the method is efficient and clearly superior to ILP solving, naive path ranking and dynamic programming. It was also highly competitive to labeling methods.

The same approach as in constrained shortest paths also applies to other network optimization problems with resource constraints: first solving a Lagrangean relaxation and then ranking solutions.

Since the problem is of great practical interest in operations research we decided to develop a general package that provides efficient algorithms for specific problems and is easily adapted to other problems.

This paper is organized as follows. In the next section we present the theory underlying constrained network optimization. The design of our package is discussed in Section 3. In Section 4 we describe four applications that illustrate the power and flexibility of the package.

* Partially supported by the IST Programme of the EU under contract number IST-1999-14186 (ALCOM-FT).
** Partially supported by a Graduate Fellowship of the German Research Foundation (DFG).

2 Underlying Theory

Suppose we are given a network G with n nodes and m edges and a cost function $c : E \to \mathbb{R}^+$ defined on the edges.

Many linear network optimization problems like shortest paths, minimum spanning trees or minimum cuts ask for the computation of a list of edges satisfying specific constraints (to form a path, a spanning tree etc.) such that the sum of the edge costs is minimized.

Using 0-1-variables z_e for each edge in the graph this can be written as an integer linear program (ILP) as follows:

$$\begin{aligned} \min \ & \sum_{e \in E} c_e z_e \\ \text{s.t.} \ & z \in S \end{aligned}$$

where $z \in S$ abbreviates the specific constraints (path, spanning tree, etc).

For many network optimization problems of this form there exist efficient algorithms to solve the problem in polynomial time $O(p(n,m,c))$ (like in the case of shortest paths, minimum spanning trees etc.).

If we also have a resource function $r : E \to \mathbb{R}^+$ defined on the edges and impose the additional constraint that the resource consumption of our optimal solution should not exceed a given limit λ we get a (resource) constrained network optimization problem of the form:

$$\begin{aligned} \min \ & \sum_{e \in E} c_e z_e \\ \text{s.t.} \ & z \in S \\ & \sum r_e z_e \leq \lambda \end{aligned}$$

The additional resource constraint usually makes the problem *NP*-hard as it is for example the case for constrained shortest paths or constrained minimum spanning trees (see [GJ79]).

For the constrained shortest path problem, Handler and Zang [HZ80], Beasley and Christofides [BC89] and Mehlhorn and Ziegelmann [MZ00] proposed primal-dual methods to solve the Lagrangean relaxation of the problem. The key idea there is to relax the resource constraint in Lagrangean fashion, moving it into the objective function such that for a fixed Lagrange multiplier we only have to solve a corresponding unconstrained problem for modified costs.

We now briefly review the method of Mehlhorn and Ziegelmann [MZ00] in the general setting:

We introduce integer variables x_{sol} for each possible solution (e.g. path, spanning tree, etc.) of our problem. So our problem can be stated as:

$$
\begin{aligned}
\min \quad & \sum c_{sol} x_{sol} \\
\text{s.t.} \quad & \sum x_{sol} = 1 \\
& \sum r_{sol} x_{sol} \leq \lambda
\end{aligned}
$$

If we drop the integrality constraint and set up the dual problem we get:

$$
\begin{aligned}
\max \quad & u + \lambda v \\
\text{s.t.} \quad & u + v r_{sol} \leq c_{sol} \quad \forall sol \\
& v \leq 0
\end{aligned}
$$

Now we have only two variables albeit a possibly exponential number of constraints.

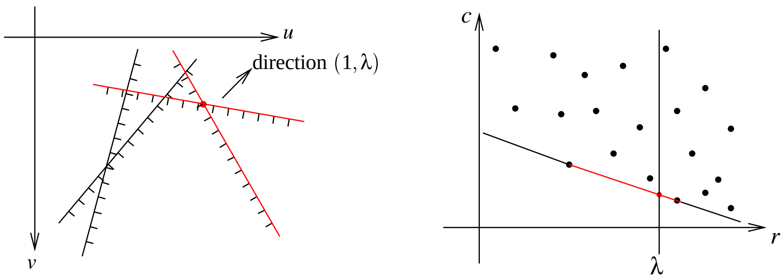

Fig. 1. (Left) Find the maximal point in direction $(1,\lambda)$ in the halfspace intersection. (Right) Find line with maximal c-value at $r = \lambda$ which has all points above or on it.

The (implicitly given) constraints can be interpreted as halfspace equations, so the problem is to find the maximal point in direction $(1,\lambda)$ in the halfspace intersection (see left part of Figure 1).

Since we have a bicriteria problem, each solution has a cost and a resource value hence can be interpreted as a point in the r-c-plane. Thus the optimal solution of the relaxation is the line with maximal c-value at the limit line $r = \lambda$ having all points above or on it (see right part of Figure 1).

The hull approach [MZ00] now finds this optimal line by computing extreme points, i.e. points on the lower hull. Starting with the segment defined by the minimum cost and the minimum resource point we want to compute the extreme point in its normal direction. The invariant is that we maintain the tentatively optimal hull segment of the extreme points seen so far. This is done in time $O(p(n,m,c))$ per iteration by solving unconstrained problems with scaled costs[1] $\tilde{c}_e = c_e - v r_e$.

[1] v is the slope of the current segment

Mehlhorn and Ziegelmann [MZ00] showed that for integral costs and resources (C and R denoting the maximum cost and resource of an edge, respectively) the number of iterations in the hull approach is $O(\log(nRC))$. Their proof also extends for the general constrained network optimization case, so we can conclude that the hull approach computes the optimum of the relaxation in $O(\log(nRC)p(n,m,c))$ time which is polynomial in the input.

Solving the relaxation with the hull approach we obtain a feasible solution giving an upper bound for the problem whereas the optimal solution of the relaxation provides a lower bound. A duality gap may exist since the upper bound, the line found by the hull approach and the limit line define an area where the true optimal solution may lie (see Figure 2).

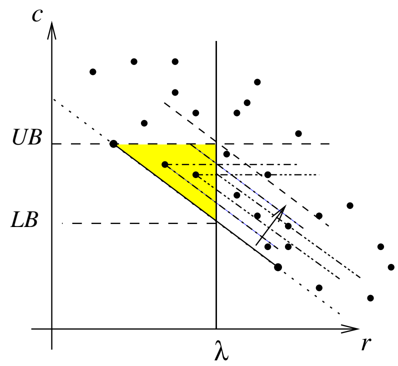

Fig. 2. Closing the gap between upper and lower bound.

We may close this gap by a solution ranking procedure, we enumerate solutions by increasing scaled cost. Lawler gave a general method for ranking discrete optimization problems [Law72] (there exist special algorithms for path ranking [JM99, Epp99] and spanning tree ranking [KIM81, Epp90]).

In the gap closing step we rank the solutions with respect to the scaled costs corresponding to the optimum of the relaxation. This can be viewed as sweeping the area in which the optimal solution may lie with the line found by the hull approach. We update the bounds during the ranking process and may stop when we have swept the promising area (see Figure 2).

Here is where *NP*-hardness comes in, we cannot give a polynomial bound on the number of solutions that have to be ranked. There also is no approximation guarantee for the bounds resulting from the relaxation [MZ00]. However, experiments for the constrained shortest path problem [HZ80, BC89, MZ00, Xue00] show that the bounds are usually very good and hence lead to an efficient gap closing step (see also Table 3)[2].

[2] A straightforward adaption of the hull approach for a single resource enables the computation of the whole lower hull of the underlying problem in $O((nRC)^{2/3})$ time which provides an idea of how good the bounds are for different resource limits.

The hull approach also works for multiple resources (i.e. having k resource functions and k resource limits)[3]. The gap closing step via solution ranking also extends.

We obtain a general 2-phase method for constrained network optimization problems for an arbitrary number of resources: first solve the Lagrangean relaxation and then close the duality gap via solution ranking.

Apart from the hull approach[4] there also exist other methods for solving the Lagrangean relaxation: In the single resource case, we can also use binary search on the slopes [Xue00, MZ00]. Beasley and Christofides [BC89] proposed a subgradient procedure that approximates the optimum of the relaxation (also for multiple resources).

In the next section we describe a software package that offers all these relaxation approaches and provides a generic framework for the 2-phase method for constrained network optimization.

3 Design of the CNOP Package

We describe CNOP, a Constrained Network Optimization Package. We implemented the package in C++ using LEDA [MN99, MNSU].

The main function of the package is

```
RESULT cnop(G,s,t,cost,resource,upperbound, netopt,ranking);
```

Here G is the graph of the underlying network problem, s and t are nodes, `cost` is the cost function defined on the edges, `resource` is the resource function(s) defined on the edges and `upperbound` is the resource limit(s).

`netopt` is a function used as core function by the relaxation methods. It is of general form

```
list<edge> netopt(G,s,t,cost,resource, scalevector, c, newpoint);
```

where `scalevector` specifies how to scale the new weight function for the network optimization. The result of the scaled optimization is returned as a list of edges, its value in `c` and the cost and resource value of the solution is returned in `newpoint`.

When the user has specified this function and the desired method to solve the relaxation (hull approach, binary search and subgradient procedure in the single resource case and hull approach and subgradient procedure in the multiple resource case), the optimal value of the relaxation is returned.

To close the possible duality gap the user now has to specify the function `ranking` which should be of the form

```
bool ranking(G,s,t,cost,resource,upperbound, scaledcost,
             scalevector, UB, LB, UBsol);
```

[3] However, [MZ00] could not show a polynomial bound on the number of iterations of the hull approach for multiple resources.

[4] The method of Handler and Zang [HZ80] for the single resource case is very similar to the hull approach.

where `scaledcost` is the scaled cost function leading to the optimum of the relaxation, `UB` and `LB` are upper and lower bounds and `UBsol` the solution resulting from the relaxation. Now the `ranking` function should rank the solutions with respect to weight function `scaledcost` and update the bounds `UB, LB` and best feasible solution `UBsol` until the optimum is found or unfeasibility is detected.

The user also may provide a special function testing the feasibility of a solution and thus may also incorporate lower bounds on the resource consumption, etc.

Since the gap closing step may take a long time depending on the scaled costs it is possible to specify break criteria, e.g. the number of solutions to be ranked or difference between upper and lower bound.

This provides the generic concept for constrained network optimization as discussed in Section 2. The `cnop` function returns a list of edges specifying the optimal solution or reports infeasibility. If the ranking was aborted before closing the duality gap we return lower and upper bound with the best feasible solution.

At the moment our package is tested for `gcc 2.95.2` under Solaris but we plan to support other compilers and platforms in the future as well as providing CNOP as a LEDA extension package. The hull approach for multiple resources makes use of either the implementation of the d-dimensional convex hull algorithm [CMS93] in the Geokernel LEP [MNS] or the CPLEX callable library [CPL], the hull approach for the single resource case works without additional software.

3.1 Special Case: Constrained Shortest Paths

The constrained shortest path problem is to find a minimum cost path between two nodes that satisfies the resource limit(s).

Previous work on constrained shortest paths can be categorized into three different approaches: dynamic programming methods [Jok66], labeling methods [AAN83, DDSS95] and 2-phase methods [HZ80, BC89, MZ00] first solving the Lagrangean relaxation and then closing the duality gap (see Section 2).

CNOP offers a special function for constrained shortest paths:

As core algorithm for the relaxation methods in our generic approach we use LEDA's implementation of Dijkstra's algorithm but a user also may choose an own shortest path implementation.

For the gap closing step we offer three different methods: We reimplemented the recursive enumeration algorithm of Jiminez and Marzal [JM99] which seems to perform better in practice than the theoretically best known k-shortest path algorithm of Eppstein [Epp99]. Moreover, we implemented a labeling setting algorithm with certain heuristics [MZ00] that is similar to the one proposed by Aneja et al. [AAN83] and a label correcting approach as in [DDSS95]. We also offer the dynamic programming implementation of Joksch [Jok66, BC89]. Of course it is also possible to provide an own implementation for the gap closing step.

In addition to the general framework, we also offer problem reduction methods as proposed by Aneja et al. [AAN83] and Beasley and Christofides [BC89]. Nodes and edges of the graph are removed whose inclusion in a path would force the resource consumption over the resource limit or the length over the upper bound on the optimal solution. These reductions may reduce the size of a problem considerably enabling a faster gap closing step. They can be switched on or off by the user.

So a user is able to experiment with all the known "state of the art" methods, trying out arbitrary combinations to see which setting fits best for a particular application.

The general function looks like this: (reductions are turned on, and the k-shortest path algorithm is used per default)

```
csp(G,s,t,cost,resource,upperbound);
```

Default parameters enable the user to turn on/off reductions and relaxation computation and switch between different methods (or provide new ones).

3.2 Special Case: Constrained Minimum Spanning Trees

The constrained minimum spanning tree problem is to find a spanning tree of minimal cost that satisfies the resource limit(s).

Ahuja et al. [AMO93] first reported that the Lagrangean relaxation can be solved with a subgradient procedure. Xue [Xue00] gave a binary search procedure for solving the Lagrangean relaxation in the single resource case. Using our hull approach we may solve the relaxation exactly also for multiple resources, moreover in the single resource case for integral costs and resources we get a polynomial running time for solving the relaxation.

We also offer a special function for the constrained minimum spanning tree problem. To the best of our knowledge, this is the first implementation for the constrained minimum spanning tree problem.

As core algorithm for the relaxation methods in our generic approach we use LEDA's implementation of Kruskal's minimum spanning tree algorithm. A user again may provide an own implementation.

For the gap closing step we implemented the spanning tree ranking algorithm of Katoh, Ibaraki and Mine [KIM81][5]. Other methods may be provided by the user.

Like in the constrained shortest path case there are problem reductions possible. We used ideas of Eppstein [Epp90] to also provide functions for problem reductions in the constrained spanning tree case. They can be turned on or off by the user.

The general function looks like this: (reductions are turned on, and the spanning tree ranking algorithm is used per default)

```
cmst(G,s,t,cost,resource,upperbound);
```

Default parameters enable the user to turn on/off reductions and relaxation computation and switch between different methods (or provide new ones).

4 Applications

We now discuss four applications demonstrating the flexibility and the use of our package. At the end of this section we report some running times.

[5] We will also offer the improvement of Eppstein [Epp90] in the near future.

4.1 Route planning

Route planning is a standard application of constrained shortest paths. We are given a road or train network and a source and target destination. We want to travel from source to target destination with minimal cost satisfying the resource constraint(s). Here, costs could be for example time and resource could be fuel consumption (see left part of Figure 3).

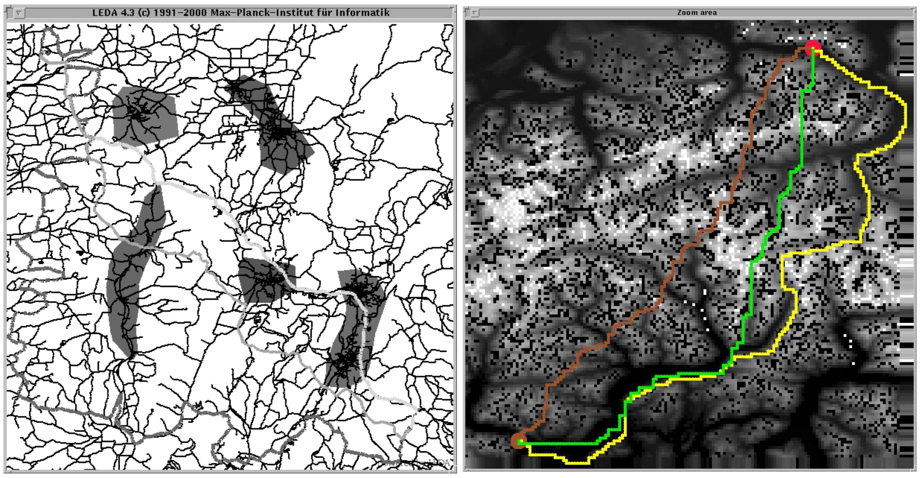

Fig. 3. Minimum time satisfying fuel constraints (congestion areas are shaded) (left) Minimum total height difference satisfying length constraint (right).The minimum cost paths are yellow, the minimum resource paths brown and the constrained shortest paths green.

Many other cost and resource models exist. For example we may want to minimize the total height difference while not travelling more than a given distance. (see right part of Figure 3). After setting up the graph and the edge costs and resources we can simply use the `csp` function of CNOP.

4.2 Curve Approximation

Piecewise linear curves (or polygons) are often used to approximate complex functions or geometric objects in areas like computer aided geometric design, cartography, computer graphics, image processing and mathematical programming. Often, these curves have a very large number of breakpoints, and for storage space or other reasons, one may be interested in an approximation of the curve using fewer breakpoints.

Dahl and Realfsen, and Nygaard et al. [DR97, NHH98, DR00] studied this problem and showed how to model this as a constrained shortest path problem:

The breakpoints $v_1, \dots, v_n$ are the nodes of the graph and for all $1 \leq i < j \leq n$ we introduce an edge (v_i, v_j). The cost of an edge is set to the approximation error introduced by taking this shortcut instead of the original curve[6]. The resource of an edge is set to 1. Now we may compute the best approximation using at most k breakpoints with our general approach (see Figure 4 for an example). Alternatively, we may also compute the minimum number of breakpoints for a given approximation error.

Since we now have identical resources and $k \leq n$, the problem is not NP-hard anymore, since the dynamic programming formulation now runs in $O(kn^2) = O(n^3)$.

Dahl and Realfsen [DR97, DR00] observed that solving the relaxation outperforms the exact dynamic programming but pointed out the problem that it cannot guarantee optimality although the optimum was often reached in their experiments. Whereas Nygaard et al. [NHH98] consider the dynamic programming method as state of the art.

Using our package, we can confirm the experiments of [DR97, DR00] for larger number of breakpoints, comparing the 2-phase approach with dynamic programming (see also Table 3). Moreover, the gap closing time is usually dominated by solving the relaxation.

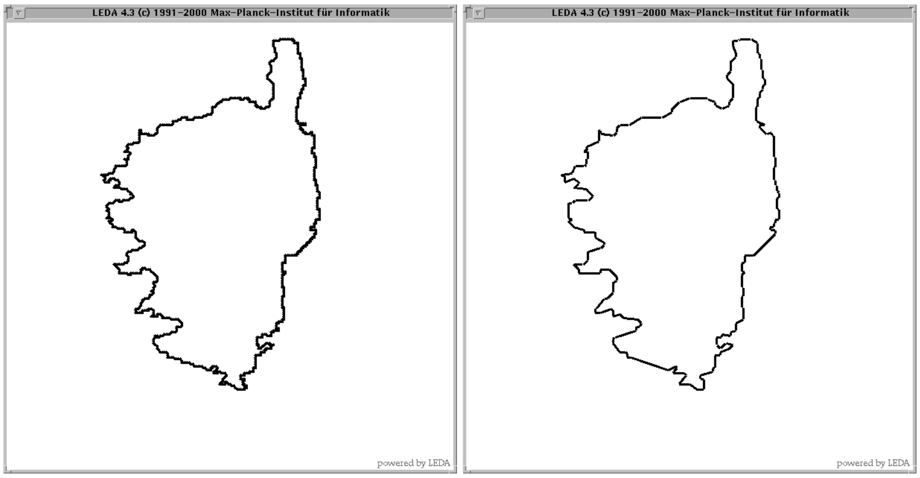

Fig. 4. Coastline of Corsica (800 points) and minimum error approximation using 200 points

There are numerous other applications of the constrained shortest path problem:

Modeling Engineering Problems as Constrained Shortest Paths Elimam and Kohler [EK97] show how to model two engineering applications as constrained shortest path problems: optimal sequences for the treatment of wastewater processes and minimum cost energy-efficient composite wall and roof structures.

[6] Different error metrics may be used.

Constrained Shortest Paths as a Subproblem There are several applications in operations research that involve constrained shortest paths as a subproblem. Most of the time these are column generation methods. Examples are duty scheduling [BBKL00], traffic routing with congestion [JMS99] and scheduling switching engines [LZ00].

Constrained geodesic shortest paths in TINs Given a triangulated irregular network (TIN) describing a terrain surface, we also may ask for the continuous (geodesic) version of the constrained shortest path problem. Now the length of a path link is its geodesic length and the resource of a path link could for example be its slope. Now we have geodesic shortest paths as a core problem. We reimplemented the approximate scheme of Lanthier, Maheswari and Sack [LMS00]. Plugging this core algorithm into our hull approach we get an approximate solution to the optimal relaxation [MSZ00].

4.3 Minimum cost reliability constrained spanning trees

The previous applications all had shortest paths as a core routine. Now we turn to a different constrained network problem, the minimum cost reliability constrained spanning tree problem that arises in communication networks: We are given a set of n stations in the plane that can communicate with each other. We now want to connect the stations, the cost of a connection might be modeled by the distance of the stations and the reliability of a connection by its fault probability. We now want to compute a minimum cost connection (spanning tree) such that its total fault probability is beyound a given limit.

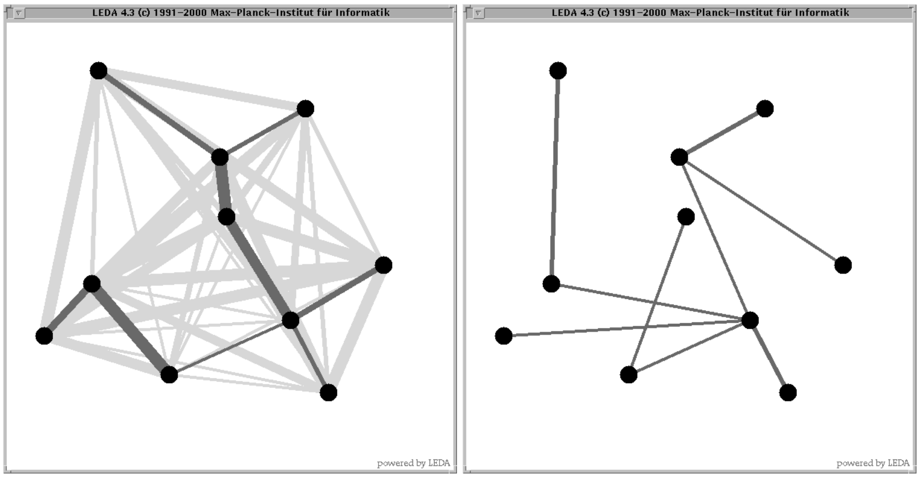

Fig. 5. Minimum cost spanning tree and Minimum cost reliability constrained spanning tree. Width of edges corresponds to fault probability.

After setting up the graph with its costs and resources we can simply use the `cmst` function of CNOP. Figure 5 gives an example.

4.4 Table layout

Consider the problem of laying out a two-dimensional table. Each cell of the table has content associated with it and we have choices on the geometry of cells. The table layout problem is to choose configurations for the cells to minimize the table height given a fixed width for the table (see Tables 1 and 2 that display excerpts of the book bestseller list in Germany). The problem is motivated by typography, both traditional print-to-paper and online web typography.

Belletristik	**Sachbücher**
Joanne Rowling: Harry Potter	Marcel Reich- Ranicki: Mein Leben
Rosamunde Pilcher: Wintersonne	Lance Armstrong: Tour des Lebens
Donna Leon: In Sachen Signora Brunetti	Florian Illies: Generation Golf

Table 1. Minimum height table without width constraints.

Anderson and Sobti [AS99] studied this problem and showed how to model it as a flow problem in a graph. Solving scaled minimum cut computations as in the hull approach they solve a relaxation.

Belletristik	**Sachbücher**
Joanne Rowling: Harry Potter	Marcel Reich- Ranicki: Mein Leben
Rosamunde Pilcher: Wintersonne	Lance Armstrong: Tour des Lebens
Donna Leon: In Sachen Signora Brunetti	Florian Illies: Generation Golf

Table 2. Minimum height table with total width $\leq$ 29 characters.

We reimplemented their graph modeling and apply our general hull scheme using the max flow-minimum cut algorithm of LEDA. After setting up the graph and the special minimum cut function we may use our package.

The hull approach gives us the optimal solution of the relaxation and a feasible solution providing an upper bound for the table layout problem. If we want to have the optimal solution we have to rank the cuts in increasing reduced cost order. This can be done with the algorithm of Hamacher et al. [HPQ84]. We plan to include an implementation of this algorithm in the future.

The symmetric problem of minimizing the table's width for a given height is also easily solved using our package. Our package would also allow to solve a 3d table layout problem where each cell also has a length, i.e. minimize height of the table given limits on the width and length of the table.

Experiments

At the end of this section we want to give an idea about the running time of different provided constrained shortest path methods on the discussed applications. Since the running time is dependent on the size and structure of the graph, the distribution of the edge costs and resources and the "hardness" of the constraint(s), we cannot give a *detailed* running time comparison of different methods. However, Table 3 gives some ideas about the efficiency of different constrained shortest path methods on various kinds of graphs[7]. We see that the 2-phase method (hull approach + gap closing) is superiour to naive approaches like ILP solving or dynamic programming. It is also faster than the labeling approach, especially for larger graphs, except in the case of road graphs where it is is slightly worse than the labeling. Moreover, the running time of the 2-phase method does not seem to depend too much on the structure of the graph but just on the size of the graph. A user can easily do a detailed running time comparison for a specific problem using the package.

5 Summary

We proposed a package for constrained network optimization problems. Even though there exist several implementations of different algorithms for the constrained shortest path problem, this is the first publicly available package that provides implementations of state of the art algorithms and allows flexible exchange of different methods.

The package also provides the first available implementation for computing constrained minimum spanning trees.

The general framework can easily be adapted to solve other constrained network optimization problems as we have seen in the section about the table layout problem.

We believe that the flexible design of the package makes it very easy to use. Moreover, it is customizable to the user's needs. Due to the practical importance of the provided methods we believe that it will be useful - at least for experimental purposes - for a large number of people working in operations research.

Future work includes implementing some other ranking algorithms and thus providing complete optimization code for other examples. We also plan to offer the package as a LEDA extension package (LEP) and support other platforms and compilers.

The CNOP package including manual and applications can be freely downloaded for non-commercial use from the URL

`http://www.mpi-sb.mpg.de/~mark/cnop`

[7] All experiments measuring CPU time in seconds on a Sun Enterprise 333MHz, 6Gb RAM running SunOS 5.7 using LEDA 4.3 and CNOP compiled with gcc-2.95.2 -O.

type	n	m	λ	OPT	UB	LB	it	rel+gc	label	DP	ILP
DEM	11648	46160	2579.5	18427	18509	18390.5	8	1.6	4.2	-	-
DEM	41600	165584	4650.8	27863	27913	27836.6	10	8.7	40.3	-	-
DEM	16384	65024	2810.5	3669	4381	3626.9	7	2.3	5.5	-	-
DEM	63360	252432	5897.1	5695	6045	5676.6	8	11.7	85.9	-	-
ROAD	24086	50826	816.9	3620	3620	3513.3	6	1.9	1.0	-	-
ROAD	77059	171536	1324.9	400	400	392.8	6	8.2	7.1	-	-
CURVE	1000	9945	300	5977.23	5977.23	5977.23	9	0.27	1.9	2.6	8.1
CURVE	5000	49945	800	72097.7	72220	72097	12	1.6	14.7	52.3	150.4
CURVE	10000	99945	2000	107208	107208	107208	13	3.3	103.4	-	-

Table 3. Running time for different constrained shortest path methods on different graph structures. The first column describes the graph type. DEM stands for digital elevation models (see Figure 3 (right)), i.e. bidirected grid graphs with cost of an edge equal to the height difference of its nodes and resource of an edge randomly chosen from [10,20]. ROAD stands for US road graphs (see Figure 3 (left)) with the cost of an edge equal to the congestion time and the resource of an edge equal to its euclidean length. CURVE stands for graphs obtained from approximating a random linear signal (see Subsection 4.2 (but here we limit the outdegree of the nodes to 10)) with cost of an edge being the approximation error and resource of an edge equal to 1. Columns 2 and 3 contain the number of nodes and edges of the graph. Column 4 contains the resource limit (which is set to 110% of the minimum resource path in the graph in the DEM and ROAD case) and Column 5 contains the optimum. The following three columns contain upper and lower bound obtained by the hull approach and the number of iterations. The last four columns contain the running time in seconds for the 2-phase approach, label correcting, dynamic programming and ILP solving. A "-" means that computation was aborted after 5 minutes.

References

[AAN83] Y. Aneja, V. Aggarwal, and K. Nair. Shortest chain subject to side conditions. *Networks*, 13:295–302, 1983.

[AMO93] R. Ahuja, T. Magnati, and J. Orlin. *Network Flows*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[AS99] R. Anderson and S. Sobti. The table layout problem. In *Proc. 15th SoCG*, pages 115–123, 1999.

[BBKL00] B. Böhringer, R. Borndörfer, M. Kammler, and A. Löbel. Scheduling duties by adaptive column generation. In *CASPT2000*, 2000.

[BC89] J. Beasley and N. Christofides. An Algorithm for the Resource Constrained Shortest Path Problem. *Networks*, 19:379–394, 1989.

[CMS93] K. Clarkson, K. Mehlhorn, and R. Seidel. Four results on randomized incremental construction. *Computational Geometry: Theory and Applications*, 3(4):185–212, 1993.

[CPL] CPLEX Optimization, Inc. *Using the CPLEX callable library*. `http://www.cplex.com`.

[DDSS95] J. Desrosiers, Y. Dumas, M. Solomon, and F. Soumis. *Time constrained routing and scheduling*, volume 8 of *Handbooks in Operations Research and Management Science: Network Routing*, chapter 2.4 Constrained shortest path problems, pages 70–80. Elsevier Science, Netherlands, 1995.

[DR97] G. Dahl and B. Realfsen. Curve approximation and constrained shortest path problems. In *International Symposium on Mathematical Programming (ISMP97)*, 1997.

[DR00] G. Dahl and B. Realfsen. The cardinality-constrained shortest path problem in 2-graphs. *Networks*, 36(1):1–8, 2000.

[EK97] A. Elimam and D. Kohler. Two engineering applications of a constrained shortest path model. *European Journal of Operational Research*, 103:426–438, 1997.

[Epp90] D. Eppstein. Finding the k smallest spanning trees. In *Proc. 2nd Scandinavian Worksh. Algorithm Theory (SWAT)*, LNCS 447, pages 38–47. Springer, 1990.

[Epp99] D. Eppstein. Finding the k shortest paths. *SIAM Journal on Computing*, 28(2):652–673, 1999.

[GJ79] M. Garey and D. Johnson. *Computers and Intractibility: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, 1979.

[HPQ84] H. Hamacher, J. Picard, and M. Queranne. On finding the k-best cuts in a network. *Operations Research Letters*, 2(6):303–305, 1984.

[HZ80] G. Handler and I. Zang. A Dual Algorithm for the Constrained Shortest Path Problem. *Networks*, 10:293–310, 1980.

[JM99] V. Jimenez and A. Marzal. Computing the k shortest paths. A new algorithm and an experimental comparison. In *Proc. 3rd Workshop on Algorithm Engineering (WAE99)*, LNCS 1668, pages 15–29. Springer, Berlin, 1999.

[JMS99] O. Jahn, R. Möhring, and A. Schulz. Optimal routing of traffic flows with length restrictions in networks with congestion. Technical report, TU Berlin, 1999.

[Jok66] H. Joksch. The Shortest Route Problem with Constraints. *Journal of Mathematical Analysis and Application*, 14:191–197, 1966.

[KIM81] N. Katoh, T. Ibaraki, and H. Mine. An algorithm for finding k minimum spanning trees. *Siam Journal on Computing*, 10(2):247–255, 1981.

[Law72] E. Lawler. A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. *Management Science B*, 18:401–405, 1972.

[LMS00] M. Lanthier, A. Maheshwari, and J.R. Sack. Approximating shortest paths on weighted polyhedral surfaces. *Algorithmica*, 2000. preliminary version in SCG97.

[LZ00] M. Lübbecke and U. Zimmermann. Computer aided scheduling of switching engines. In *CASPT2000*, 2000.

[MN99] K. Mehlhorn and S. Näher. *The LEDA Platform for combinatorial and geometric computing*. Cambridge University Press, 1999.

[MNS] K. Mehlhorn, S. Näher, and M. Seel. *The Geokernel LEP User Manual Version 2.1*. Max-Planck-Institut für Informatik. `http://www.mpi-sb.mpg.de/LEDA/friends/dd_geokernel.html`.

[MNSU] K. Mehlhorn, S. Näher, M. Seel, and C. Uhrig. *The LEDA User Manual*. Max-Planck-Institut für Informatik. `http://www.mpi-sb.mpg.de/LEDA`.

[MSZ00] K. Mehlhorn, J.R. Sack, and M. Ziegelmann. Constrained geodesic shortest paths. manuscript, 2000.

[MZ00] K. Mehlhorn and M. Ziegelmann. Resource constrained shortest paths. In *7th Annual European Symposium on Algorithms (ESA2000), LNCS 1879*, pages 326–337, 2000.

[NHH98] R. Nygaard, J. Husøy, and D. Haugland. Compression of image contours using combinatorial optimization. In *Int. Conf. on Image Processing (ICIP)*, 1998.

[Xue00] G. Xue. Primal-dual algorithms for computing weight-constrained shortest paths and weight-constrained minimum spanning trees. In *19th IEEE International Performance, Computing, and Communications Conference (IPCCC 2000)*, pages 271–277, 2000.