Approaches for a Single-Source Network-Loading Problem

Ivana Ljubic[1], Peter Putz[1], I. Rodríguez[2], J.J. Salazar[2]

[1] University of Vienna, Austria
ivana.ljubic@univie.ac.at ; p.putz@univie.ac.at
[2] University of Tenerife, Spain
irguez@ull.es ; jjsalaza@ull.es

1 Introduction

We are working on the following problem. Let us consider a set D of customers which require a demand from a single source r. In addition, there are a set W of intermediate points, and all together $V := \{r\} \cup D \cup W$ is the set of nodes. Some pair of nodes can be linked with cables, thus conforming an undirected graph $G^u = (V, E)$. Each cable will be represented by two directed arcs, thus we work with a directed graph $G = (V, A)$. Each edge $e \in E$ has a length l_e and each destination $v \in D$ has a demand d_v that must arrive to v from r. There is a set of cable types N, one of which ($n \in N$) may be installed in an edge with a cost c_n and with a capacity u_n.

The *Single Source Network Loading* (SSNL) problem asks for

- an installation of cables on the edges, such that
- flows from the root to all customers can be routed simultaneously,
- satisfying the demands,
- not violating the arc-capacities, and
- the sum of all cost shall be minimal.

There are several previous publications on a similar problem: Barahona (1996), Gabrel, Knippel and Minoux (1999), Günlük (1999), Berger et al. (2000), Randazzo, Luna and Mahey (2001), Garg et al. (2001), Atamürk (2002), Costa (2005), Avella, Mattia and Sassano (2007), Raack, Koster and Wessäly (2007), Costa, Cordeau and Gendron (2007), ... On the problem itself we are aware of two articles: Gupta et al. (2007) and Salman, Ravi and Hooker (2008).

2 Mathematical models

By using $x_a^n = 1$ if and only if cable type n is installed on arc a, and $f_a \geq 0$ representing the amount of flow on arc $a \in A$, then a single-commodity flow formulation (SCF) is:

$$\min \sum_{a \in A} \sum_{n \in N} l_a c_n x_a^n \tag{1}$$

$$\sum_{a\in\delta^+(v)} f_a - \sum_{a\in\delta^-(v)} f_a = \begin{cases} -d_v, & v \in D \\ \sum_{i\in D} d_i, & v = r \\ 0, & \text{others} \end{cases} \qquad \forall v \in V \tag{2}$$

$$f_a \le \sum_{n\in N} u_n x_a^n \qquad \forall a \in A \tag{3}$$

$$\sum_{n\in N} x_a^n \le 1 \qquad \forall a \in A \tag{4}$$

$$x_a^n \in \{0,1\}, f_a \ge 0 \qquad \forall a \in A,\ \forall n \in N \tag{5}$$

We have developed other multi-commodity flow models and their associated Benders' decomposition approaches. In particular we have put a lot of emphasis in implementing tricks to speed up the Benders' decomposition approach for a model, with stronger inequalities and efficient cut separation algorithms.

3 Preliminary Computational Results

In our talk we will present the results of our experiments based on instances from SNDlib 1.0 (http://sdnlib.zib.de). The following table shows preliminary results on some instances comparing our codes based on a single-commodity flow formulation (SCF), a multi-commodity flow formulation (MCF), a multi-cable multi-commodity flow formulation (MMCF) and the associated Benders' decomposition (BEND) strengthened with some additional valid inequalities. For each model the table shows the "gap" at the root node, and the total "time" to optimally solve the problem.

name	$\lvert V\rvert$	$\lvert E\rvert$	$\lvert N\rvert$	SCF gap	SCF time	MCF gap	MCF time	MMCF gap	MMCF time	BEND gap	BEND time
cost266	37	57	3	57.95	7.55	24.33	184.27	14.88	1356.53	14.15	-
dfn-bwin	10	45	55	42.92	0.31	7.28	2.53	0.11	226.83	0.01	424.73
dfn-gwin	11	47	2	32.53	0.97	27.64	9.64	13.02	1.48	3.51	7.88
di-yuan	11	42	6	24.33	0.48	15.21	6.03	12.04	8.34	7.48	71.97
NewYork	16	49	2	34.12	0.22	13.20	2.75	10.00	4.42	3.10	17.28
nobel-Ge	17	26	40	46.34	2.73	11.83	0.95	10.00	50.25	5.80	3621.25
nobel-US	14	21	40	7.79	0.23	5.97	2.97	3.93	73.27	0.44	1065.92
Norway	27	51	2	54.73	13.38	13.62	137.59	8.74	311.02	7.84	-
pdh	11	34	3	37.81	0.17	34.95	1.70	6.07	0.56	1.49	4.77
ta2	65	108	11	93.15	783.41	0.00	0.75	0.00	16.05	0.00	1001.44

As expected, BEND is the model providing the best lower bound at the root node of the branch-and-bound search. However, our implementation of BEND is also the slower one. Our current work is trying to add tricks to speed up these codes. Hopefully we will present better results during the GOM2008 conference.