Winter semester 2013/14 Research Institute for Discrete Mathematics
Prof. Dr. S. Held University of Bonn

Linear and Integer Optimization

Exercise Sheet 10

Exercise 10.1: Let $\mathcal{F} = \{x \in \mathbb{Z}^n : Ax \leq b; x \geq 0\}$ with $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$. Furthermore, let $F : \mathbb{R}^m \to \mathbb{R}$ be a function that is *superadditive*, i.e. $F(a_1)+F(a_1) \leq F(a_1 + a_2)$ for all $a_1, a_2 \in \mathbb{R}^n$, *non-decreasing*, i.e. $F(a_1) \leq F(a_2)$ for $a_1 \leq a_2$, and that fulfills $F(0) = 0$.

1. Prove that the inequality
$$\sum_{j=1}^{n} F(A_j)x_j \leq F(b)$$
holds for all $x \in \text{conv}(\mathcal{F})$, where A_j is the j-th column of A. (4 Points)

2. Conclude, that the following inequalities hold for all $x \in \text{conv}(\mathcal{F})$:
$$\sum_{j=1}^{n} \lfloor u^\intercal A_j \rfloor x_j \leq \lfloor u^\intercal b \rfloor$$
for all $u \in \mathbb{R}^m_{\geq 0}$. (1 Point)

Exercise 10.2: Prove that any unimodular matrix arises from the identity matrix by unimodular transformations. (5 Points)

Exercise 10.3: Prove the integral version of Carathéodory's theorem. For any pointed rational polyhedral cone $C \subset \mathbb{R}^n$, any Hilbert basis $\{a_1, \ldots, a_l\}$ of C and any integral point $x \in C \cap \mathbb{Z}^n$, there are $2n - 1$ vectors from the Hilbert basis such that x is a non-negative integral combination of these vectors. (5 Points)

Submission deadline: Tuesday, 7.1.2014, before the lecture.

Programming Exercise 3:
Implement the branch-&-bound algorithm for the MAXIMUM-WEIGHT-STABLE-SET-PROBLEM that is defined as follows. Given a graph G and weights on the vertices $\alpha : V(G) \to \mathbb{N}$, we are looking for a stable set $S \subseteq V(G)$ of maximum weight $\sum_{v \in S} \alpha(v)$. It should be modeled by the following ILP:

$$\max \sum_{v \in V(G)} \alpha(v) x_v \tag{1}$$

$$s.d. \quad x_v + x_w \leq 1 \qquad \forall \{v, w\} \in E(G) \tag{2}$$

$$x_v \in \{0, 1\} \qquad \forall v \in V(G) \tag{3}$$

As an LP solver you must use the academically free program **QSopt** through the API in `lp.h` that is available on the web-site. To make the implementation easier, you find a program that

1. Reads an instance,
2. creates the above LP-relaxation using the API in `lp.h`,
3. solves it and prints the solution vector to the console.

(see `http://www.or.uni-bonn.de/~held/lpip/1314/mss.zip`). The ZIP file contains also test instances. **Read the README file for further information!**. The **32-bit** program compiles under Linux or Windows/Cygwin (`gcc -m32 ...`)! For compiling type 'make' in the extracted directory 'mss'.
Note that the matrix A is usually sparse, i.e. most coefficients are zero. Thus, in `lp.h` new rows/constraints are always defined by their non-zero entries. The corresponding functions in `lp.h` are commented and their use becomes clear in mss.c.
As you observed on the last exercise sheet, the LP relaxation has a large integrality gap, which is problematic for branch-&-bound. Thus you should first try to tighten the gap in the root LP, by adding clique inequalities (Exercise 9.4,2). To this end you should implement a simple greedy algorithm that starts with $C = \{v\}$ for a $v \in V(G)$ and adds vertices $w \in V(G) \setminus C, C \subseteq \delta(w)$, with a large value x_w to C. This should be started iteratively with different vertices $v \in V$ until all vertices are part of some (inclusion-wise) maximal clique. You should iterate the two steps

- solving the root lp and
- adding clique inequalities

until no more clique inequalities are found before starting branch-&-bound. The algorithm should write

1. the value of the root LP without clique inequalities,
2. the value of the root LP with clique inequalities, and
3. the value and vertex indices of a maximum-weight stable set S

to the console. (20 Points)
Submission deadline of the programming exercise: Tuesday, 14.01.2014, before the lecture.