Prof. Dr. Rüdiger Ehlers Francisco Palau, M.Sc.

Applied Computational Engines 2018 – Assignment Sheet 8

Due: Monday, **11th June 2018**, 10 am

Please indicate your **name** and **email address**. You can work in **groups** of up to **three** students. Only one submission per group is necessary. However, in the tutorials every group member must be able to present the solutions to each problem solved by your group.

Please submit your solutions either

- by e-mail to `fpalau@uni-bremen.de` **and** `rehlers@uni-bremen.de`, or
- on paper in **letter box 52** (Francisco Palau-Romero) on floor 6 of the MZH building.

Note that you will need 50% of the points on all exercise sheets in order to take the "Fachgespräch" OR the oral exam. We may ask you to present your solutions in the tutorial, especially if you work in a group. We aim for asking everyone taking part in the course to present at least twice during the course of the semester.

Mixed-integer linear programming in practice (25 pts.)

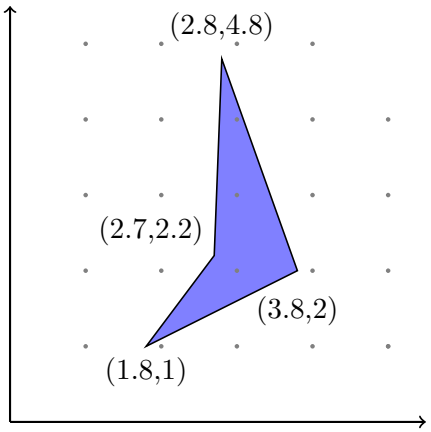
Reconsider the linear programming problem from the previous problem sheet and extend it to a mixed-integer linear programming problem as follows.

The factory manager is not completely sure if the analysis of the recycling materials is 100% precise. If the final alloy consists of more than 20% of a single recycling material, then there is a risk that this reduces the quality of the resulting alloy, and the expected cost of this glitch is \$ 20/kg per excessively used recycling material type.

So for optimization, the factory manager wants you to add \$20 to the computed cost of the alloy for every recycling material type that makes up a fraction of more than 20% in the final alloy.

Non-convexity in LP and ILP solving (15+10 pts.)

Can you build LP and/or ILP instances with sets of variables containing *at least* x and y such that exactly the valuations to x and y that can be extended to a solution to the complete LP or ILP instance fall into the following set of (partial) solutions?



The blue area in the middle is the set of solutions and the vertices of the area are labelled by their coordinates in the graph.

Solve this problem for both the 1) ILP (Integer Linear Programming) and 2) LP (Linear Programming) cases.

Whenever for one of these two cases, you find the set of (x, y) value combinations to be representable as a constraint system, please give such a system. If they are not, state why the state set is not representable.

Note: In order to encode a constraint that in a graph such as the one above, the point (x, y) lies **above** a line from (x_1, y_1) to (x_2, y_2) (for $x_2 > x_1$), you can use the following inequality:

$$y \cdot (x_2 - x_1) - x \cdot (y_2 - y_1) \geq y_1 \cdot (x_2 - x_1) - x_1 \cdot (y_2 - y_1)$$

Challenge & Bonus Problem: Cleaning up a Mess (50 pts.)

Due date for the challenge problem: 18th June, 8:00am

Somebody made a mess and your task is to use pseudo-Boolean programming and/or ILP solving to clean it up.

You will find a couple of tiles of a map on:

`http://motesy.cs.uni-bremen.de/ace2018/data/Bremen/`

We know that the map that they form has three rows of tiles and three columns of tiles. If you just attach them row-by-ro and ordered by file name, you will get the following image:

This does not look quite right – whoever assigned these files their names (`a.png` to `i.png`) made a mess!

Your task is to use pseudo-Boolean programming and/or ILP solving to find out the correct order of the tiles. The starting point is an analysis of how well each pair of tiles fits together. If the left-most pixels of a tile are identical to the right-most pixels of another tile, they fit together perfectly (horizontally). Likewise, if the bottom-most pixels of a tile are identical to the top-most pixels of another tile, they fit together perfectly (vertically). Unfortunately, we pretty much never have such perfect matches (for example due to diagonal streets), but we can accumulate the color differences between the adjacent pixels when putting two tiles next to each other. The program `countTransitionCost.py` supplied in the folder mentioned above shows how this can be done. Its output is also given in the file `lrud.txt`. The lines of the file look as follows:

```
9 d.png f.png LR
20 d.png f.png UD
15 d.png c.png LR
9 d.png c.png UD
7 d.png g.png LR
8 d.png g.png UD
8 d.png b.png LR
11 d.png b.png UD
5 d.png e.png LR
......
```

For example, the first line states that putting tile `d` left (LR) of tile `f` leads to a color difference of 9 along the edge between the tiles. Likewise, putting tile d *above* tile `f` leads to a color difference of 20 (UD).

The text file contains all possible tile combinations. Please employ a pseudo-Boolean programming and/or ILP solver to find an 3x3 arrangement of the tiles such that the sum of all placement costs are minimized. The solution to this problem should be the most likely correct arrangement of the tiles.

- You get 30 points for correctly using either a pseudo-Boolean programming solver or an ILP solver to find the correct arrangement of the tiles.
- If you try both solver types and compare the solving performance, you get the remaining 20 points.

In both cases, also try your solution on a map of London with 4 columns and 3 rows that you get under `http://motesy.cs.uni-bremen.de/ace2018/data/London/`. Do you run into computation time problems?

We need to know what the variables that you use represent, what constraints you use, the actual encoder program, and the best solution(s) that you could find when using a pseudo-Boolean problem solver or ILP solver.

There is no need to write a program that builds an image with the all tiles in the right places!

(Tiles are taken from `openstreetmap.org`)