Discrete Optimisation
Project 2: Energy-Aware Production Planning

Estimated duration for the first part: 3-5 hours

Estimated duration for the second part: 10-15 hours

22nd November 2016

Flexibility in the electricity market is currently a hot topic, as it might be a way to reduce the total energy costs within a plant, and thus to improve its competitiveness. This notion of "flexibility" is based on the fact that the electricity prices are extremely volatile due to renewable sources of energy: when the wind is strong and the sun shines unclouded, the renewable production is very high and the nuclear power plants produce the same quantity as before, hence the costly gas plants are shut down, and the electricity price decreases.

To lower their costs, industrial sites want to adapt their production planning to these high variations: the most energy-intensive products should be produced when the electricity prices are low, while the plant could be shut down when the prices are too high. (This is one kind of flexibility; see [2], Section 1.2, for a longer introduction to the topic.) Not all kinds of industrial processes are candidates, due to their constraints: for example, in the metallurgy industry, if an electrolysis bath is shut down, its temperature decreases, and the ore being melted may become solid and seriously damage the installations.

The paper industry is one good candidate for flexibility, due to the structure of their plants. Cellulose is extracted from wood by chemical processes. The fibres then go through pulpers: these machines produce *pulp* by mixing the cellulose fibres with large quantities of water; this process is rather energy-intensive. The pulp is fed into paper machines that make the needed kind of paper (texture, thickness, etc.) and dry it; this process is also quite energy-intensive. For each kind of paper, the machine must be tuned to the new parameters by hand, which may take some time before reaching the required quality. Finally, the paper rolls are converted into the final products (paper sheets, kitchen rolls, etc.).

As the order book of the plant is made of multiple products at various times, and that the pulpers are oversized, the main process to optimise is the use of the paper machine: when should each kind of paper be produced?

Remark. This project is inspired by the InduStore research project and a case study performed in a Belgian paper mill. The problem description is very accurate with respect to the issues encountered at this industrial site, except for the uncertainty about electricity prices.

A more complete description of the paper industry is provided in [1], Section 1.

1 Statement

The goal of this project is to schedule the production of different kinds of paper on the machine while minimising the total production costs (due to energy consumption and trashed paper).

This scheduling has some constraints to respect, but transitions between types of paper are very costly, as a large quantity of paper must be thrown away. The loss of production is modelled as a cost. Some transitions are very costly, and are thus forbidden.

Due to the required tuning of the paper machine after a transition, a type of paper must be produced for a sufficiently long period of time; if these constraints were not respected, the complex transient effects of tuning must be implemented, and very little paper of the required quality would be produced.

The paper mill has to respect an order book. It indicates when what quantity of each kind of paper must be produced. This order book is strict: no delays are allowed. The quantities are expressed as the total number of hours the paper machine must be producing the given kind of paper (i.e., if the order book indicates that 54 units of paper $\mathcal{G}$ must be produced for the 79th hour, then the paper machine has to produce the paper $\mathcal{G}$ for 54 hours during the 79 first hours of the scheduling horizon).

The pulp storage should not be considered in your model: due to the oversizing of the pulpers, pulp will always be available in sufficiently large quantities and can be produced at the optimum time (i.e. pulping is already optimised to lower the electricity costs). However, the produced paper is stored once it is produced; the stock decreases when the delay of an entry in the order book expires; the stock has no maximum.

The paper mill is not obliged to produce paper at each time step. There is no transition cost between when the plant was turned off and when the production starts again. Turning down the mill does not allow to break the minimum production duration constraint (for example, if the paper machine has produced soft paper for ten hours and that type of paper has to be produced for at least twelve hours in a row, the machine cannot stop before two hours, i.e. before the minimum production time is completely elapsed). At the beginning of the scheduling horizon (i.e. the first time step), the mill is supposed to start (there is no transition cost, but the minimum production duration of the type of paper must be enforced).

2 Available data

An instance to solve is made up of these parts:

- the scheduling horizon, in hours;
- the number of type papers to produce (they are numbered starting at 1);
- an order book: an entry consists in the number of hours a type of paper must be produced, and when this quantity must be produced;
- transition costs: a matrix giving the cost to pay when the paper machine goes from one type of paper to another;
- forbidden transitions: a list of transitions between types of paper which are not allowed;
- minimum production time for a type of paper: the minimum number of hours the paper machine must produce a given type of paper;
- electricity consumption per type of paper (in kWh/h);
- electricity prices: the price per kWh each hour of the scheduling horizon.

Each instance is made up of a total of six CSV files. The scheduling horizon and the number of types of paper are implicitly given in these files. The row and/or column numbers correspond to the index of the type of paper or the hour, as described in the following.

- The order book: each row corresponds to a type of paper (from 1 to P), each column to one hour (from 1 to T).
- The transition costs: each row corresponds to a type of paper (from 1 to P).
- The forbidden transitions: each row corresponds to one transition, where the old type of paper is stored in the first column and the new one in the second column. The types of paper use the same indices as the other files (from 1 to P).
- The minimum production times: each row corresponds to a type of paper (from 1 to P).
- The energy consumptions: each row corresponds to a type of paper (from 1 to P).
- The electricity costs: each row corresponds to one hour (from 1 to T at least: there may be more values than required).

3 Challenge

The second part of this project is organised as a challenge, where the goal is to find the best possible solution for a series of instances. This set of instances will available on the project's web page in due time.

4 Instructions

The project must be carried out by groups of two. It is divided into two parts.

1. The first one is about mixed-integer linear modelling. You are expected to hand in a **brief** report (PDF format, maximum six pages) on or before Friday, 25 November 2016 (23:59:59) on the submission platform. It should contain a description of your complete model, with the variables, the objective function, and the constraints.
   The best way of providing useful explanations is to use lists, and not large blocks of text; related constraints are better presented in contiguous groups.

2. The second one is about the actual implementation. You are expected to hand in your code (ZIP or 7Z archive) on or before Friday, 16 December 2016 (23:59:59) on the submission platform, along with a short report detailing your progress since the model (maximum ten pages, included in the archive). All groups have to submit at least one solution for the challenge before the project's deadline (see Section 3).
   Your code must contain at least one algorithm for solving instances, one being based on a MILP model; you must then implement techniques to improve the total solving time with respect to the basic model. These might include formulation improvements (such as bound strengthening), cutting planes, constraint programming, local search, heuristics to find a good first solution. [1] presents an example of constraint programming and local search (with ideas that might be used to improve your formulation). For example, you might write a heuristic to find good solutions from scratch and use it to bootstrap your MILP model (one algorithm merging the two techniques); or you might write a MILP model and a constraint programming model, none being coupled (two algorithms).
   The choice of tooling is free, but your code has to work on the reference machine (Thalès). Please provide an easy-to-use script that installs your dependencies, if any, and compiles your code, plus a second script that runs your code on the instance indicated by the first argument. Examples of useful tools: Julia and JuMP, Convex.jl; Java and Gurobi's or CPLEX' API; C++ and Gurobi's, CPLEX', or SCIP's API, Google OR-Tools; Scala and OscaR.
   Particular attention should be paid to *efficiency* so that you get very good solutions within the time limit of five minutes, even for large and complex instances.

The presentations will take place on Friday, 23 December 2016. Each group will informally present their work, followed by questions; the total duration (presentation and questions) should be around ten minutes. The main points should be the algorithm(s) you developed, and how you improved the solving times with respect to a basic MILP model (acknowledging the worsened costs if you no more solve the problem to optimality); you should give a short demo of your developments. You should also discuss the performance of your code, which can be done by generating new instances on your own.

The evaluation criteria are the following:

- for the optimisation models:
  - model adaptation to the described situation
  - linearity of both the constraints and the objective function
  - links between the variables
  - quality of the report (including the language quality and organisation)
- for the implementation:
  - concordance with the optimisation model in the report
  - quality of the code (including clarity and brevity of the code)
  - improvement in the solving time to optimality and/or in the quality of the solution within the time limit

References

[1] Cyrille Dejemeppe, Olivier Devolder, Victor Lecomte, and Pierre Schaus. Forward-Checking Filtering for Nested Cardinality Constraints: Application to an Energy Cost-Aware Production Planning Problem for Tissue Manufacturing. In Claude-Guy Quimper, editor, *Integration of AI and OR Techniques in Constraint Programming*, pages 108–124. Springer International Publishing, 2016.

[2] Sébastien Mathieu. *Flexibility services in the electrical system.* PhD thesis, Université de Liège, 2016.