Operations Research Course: Exam Projects

MISIC ISC612 2007

Leo Liberti
leoliberti@gmail.com

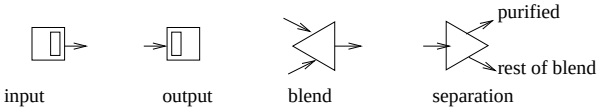
Each of the projects below has a difficulty rating out of 20 which indicates the maximum mark that can be attained on the project. You can work in groups of 1-3 people. No problem should be taken by more than one group, so please consult with each other before choosing your problem. In case of conflict, you can either fight each other to first blood or agree to split and recombine the groups so that the most motivated people for a given project are in the same group and will tackle the project at best. In case of ties, use a round of poker game, dice, a coin, forecasts on the emissions of particles of a radioactive isotope, or whatever else you may wish to call "chance". In any case, read *all* the projects before choosing one.

Warning: the projects marked "20" are really difficult and really refer to existing research project. You mustn't feel exceedingly frustrated if you don't manage to actually get good computational results or implementations that work perfectly (by contrast, I would expect good ideas and a display of good modelling techniques).

And finally, you can also propose your own projects. Write a 1-page project proposal and send it to me. I'll read it, comment it and either validate it (and give you the maximum attainable mark) or modify it, or reject it.

Every project will be supervised by one of the following researchers: Leo Liberti, Ruslan Sadykov, Laura di Giacomo (supervision will mostly be carried out by e-mail). The deadline for submission is friday 15/2/08.

1. **Separation network planning** [20]. A separation network carrying Q types of liquid consists of one input node (carrying the input blended liquid), Q output nodes (one per type of liquid), B blender nodes (which can blend two streams into one) and P separation nodes (each having one input and two output streams) capable of separating a given type of liquid from the input stream.



   The following data are known:

   - the input flow on the input node is f_0 liters per second
   - for each $j \leq Q$, the input blended liquid entering the network from the input node has a fraction $1/q_j$ of liquid of type j
   - for each $j \leq Q$, we desire an output flow of at least d_j and a purity level of at least ϑ_j at each output node j
   - for each $j \leq Q$, the unit market price of liquid j is c_j.
   - the B blender nodes simply mix the two streams together and convey the blended liquid to the output stream
   - for each $i \leq P$, separation node i separates liquid of type τ_i (an integer in $\{1, \ldots, Q\}$) with a purity $\eta_i \in [0, 1]$: this means that a fraction η_i of the liquid of type τ_i contained in the input stream of node i will go into the *purified* stream, and all the rest of the liquid types, blended with the remaining fraction $1 - \eta_i$ of liquid τ_i, will go into the other output stream

The Separation Network Planning (SCP) problem consists in maximizing the revenues (supposing each unit of output liquid is actually sold on the market) of the operation with respect to the following variables: what is the network topology (i.e. what are the interconnections between the various nodes)? what are the arc flows (i.e. what is the fraction of flow of liquid of type j in each arc)? The capacities on the pipes (the arcs) are initially supposed infinite, in the sense that the network will be built according to the capacities needed to convey the flows optimally.

Provide a mathematical programming model of the SCP problem and implement it in AMPL. Focus on the following points.

(a) What are the parameters? Give a comprehensive list.

(b) What are the variables? Give a comprehensive list.

(c) What is the objective function?

(d) What are the constraints? Give a comprehensive list.

(e) To what category does the model belong? Linear Programming (LP), Mixed Integer Linear Programming (MILP), Nonlinear Programming (NLP), Mixed-Integer Nonlinear Programming (MINLP)?

(f) According to the answer to the previous point, suggest an algorithm or a solution technique for solving the problem.

Create at least two feasible instances for this problem (i.e. create two sets of problem parameters such that the problem has at least one solution) and test your AMPL model with both (for the solution, you might need some help from me in order to choose the correct solver).

2. **MINLP solution methods** [20]. Look at the collection of MINLP problems on `http://www.gamsworld.org/minlp/minlplib/points.htm`. You have AMPL, a (continuous) NLP solver (such as `minos` or `snopt`) and a MILP solver (such as `cplex`) to work with. Put together a heuristic algorithm for solving MINLPs. It should be simple enough so that you can code it with the tools described above (AMPL and the solvers). In practice, you should find a smart way of separating the nonlinear part of the problem from the mixed-integer part so that you can solve the first with the NLP solver and the second with the MILP solver, then somehow obtain a solution for the whole MINLP from the two partial solutions. Feel free to approach me for help on the implementation once you have some good algorithmic ideas.

3. **Modelling cellular automata** [18] Read up some literature on Cellular Automata (CA) on Google (you can start from `http://mathworld.wolfram.com/CellularAutomaton.html` and `http://en.wikipedia.org/wiki/Cellular_automata`) and write a mathematical programming model that solves the following problem.

   > Given an integer T, what is the initial configuration (the status of the CA at time 0) that yields the shortest period greater than T (the number of timesteps elapsing between the first and the second display of any configuration).

   It is likely that your model will depend on the CA rule chosen. To what class (LP, MILP, NLP, MINLP) does your model belong? How can you solve it? Implement it in AMPL and test it on the CA rules 15, 30, 90.

4. **A didactical implementation of the simplex method** [14]. Write a C/C++ program that implements the simplex algorithm in the tableau version (ask me for references: it's equivalent to the matrix version that we saw in class but it's actually easier to implement because the inverse is computed implicitly), displaying a nicely formatted tableau at each iteration. Data should be read in either MPS or LP format (again, ask me for references if you choose this project).

5. **Improving UML class diagrams** [15]. Propose a mathematical programming model of the problem of rationalizing software modules in UML class diagrams. Consider a real UML class diagram with at least 30 classes (look for it on the internet or create it with software packages

such as Umbrello) and suppose you have three teams to implement the software described by the class diagram. How do you assign classes to teams so that the interactions between different teams (which are usually problematic) are minimized? Compose a set of 5 different instances and test your AMPL model on these instances. What sort of instance size (number of classes, number of class interconnections) can you solve in 1h of CPU time?

6. **Eternity II puzzle** [16]. Eternity II is an edge-matching puzzle which involves placing square puzzle pieces into a 16 by 16 grid, constrained by the requirement to match adjacent edges. The full rules are available at `http://uk.eternityii.com/about-eternity2/download/`. You aim is not to solve this puzzle. You can imagine that, if there is a 2 million dollars prize for solving this puzzle, it cannot be that easy (but still you can try!). What we ask you to do in this project, is to propose and implement a model for it which is capable of solving a smaller version of the puzzle, for example, on a 8 by 16 or 8 by 8 grid. To model the problem, you can use Integer Programming or Constraint Programming (the latter is likely to be more effective). To implement an IP model, you can use AMPL. To implement a CP model, there are several free CP solvers available (see Wikipedia page for "Constraint Programming"). Another option is, instead of providing a model, propose a heuristic algorithm for the problem (something more clever than a full search). The data will be provided to those who choose this project.

7. **Minimum Cost Flow problem** [14]. Given a flow network (a directed graph) $G(V, E)$ with source $s \in V$ and sink $t \in V$, where edge $(u, v) \in E$ has capacity q_{uv} and cost c_{uv} per unit of flow. You are required to send an amount d of flow from source s to sink t. The objective is to minimize the flow cost.

   The problem can be formulated as a Linear Program. Additionally, there are several dedicated algorithms available, for example, cycle-canceling algorithm, successive shortest path algorithm, scaling algorithm or network simplex algorithm. The first part of the project is to choose two dedicated algorithms, study them, and to produce a report, in which you should explain briefly how they work.

   The second part of the project is to implement the two algorithms chosen and compare their practical efficiency on a set of randomly generated networks. In order to save the time, you can find and use libraries which provide an implementation of the algorithms.

8. **Resource Constrained Project Scheduling Problem** [15]. We are given a project which consists of activities. Each activity has a length in days. Additionally, there are precedence relations between activities, for example, activity A cannot start before the end of activity B. The objective is to complete the project as soon as possible.

   The first part of the project is to formulate this problem as a graph problem and to propose a polynomial algorithm to solve it. What is the complexity of this algorithm? How can you name the problem in graph terms?

   The second part of the project deals with a more complex version of the problem. Assume that each activity requires a certain number of workers. For the whole project, you have a limited number of workers. Of course, every day, the activities in process should not require in total more workers than you have. Moreover, you cannot interrupt an activity once you have started it. Formulate the problem as an Integer Program and test it using AMPL on a set of randomly generated projects. How large are the instances you could solve?

9. **Solutions of Linear Complementarity Problems (LCP)** [20]. Some classical decision problems in computer science, such as the Knapsack Problem and the Satisfiability Problem can be reformulated to Linear Complementarity Problems (LCP — i.e. problems without objective function, consisting of a set of linear constraints and a bilinear constraint of the type $\sum_i x_i y_i = 0$, where $x_i, y_i \geq 0$. The aim of this project is to formulate a solution algorithm for LCP based on the simplex method, and implement this algorithm in C/C++ interfaced with CPLEX (some example code for how to interface C/C++ with CPLEX will be provided).