Department of Mathematics and Computer Science
University of Southern Denmark, Odense

February 13, 2011
Marco Chiarandini

DM826 - Modeling and Solving Constrained Optimization Problems

Obligatory Assignment 1, Spring 2011

Deadline: 1st March 2011 at noon.

Let $G = (V, E)$ be a simple graph, where $V = 1, \ldots, n$ and E are the sets of vertices and edges of G, respectively. A k-coloring of G is a mapping $\phi : V \mapsto \Gamma$ such that each vertex is assigned to a color and vertices linked by an edge have different colors. Without loss of generality $\Gamma = \{1, \ldots, n\}$. Given a k-coloring of G, we denote by C_j the color class, that is, the set of vertices assigned to color j, for each $j = 1, \ldots, n$. The task of this assignment is to find a k-coloring of G such that the color classes differ by at most one, that is, $||C_i| - |C_j|| \leq 1$, for $i, j = 1, \ldots, k$. In alternative terms, we search for a k-coloring with $\lfloor n/k \rfloor \leq |C_j| \leq \lceil n/k \rceil$ for each $j = 1, \ldots, k$. Among all k-colorings with this feature we wish to find one that minimizes k.
This variant of the graph coloring problem is relevant in applications that require load balancing, such as in parallel memory systems and scheduling. It arises also as a subproblem in timetabling. Here courses corresponds to vertices of the graph and timeslots to colors. Then, prior to room assignment, one may wish to balance the distribution of courses in timeslots so that the room assignment is facilitated.

1 Your tasks

Make sure you have read all this document including the section Practicalities before starting to work.

1. Formulate the problem as a Integer Linear Programming problem using a polynomial number of variables and constraints.

   In the model, let $K = \{1, \ldots, |V|\}$ be the set of colors. Ensure that if a color j is used then also colors $1..j-1$ are used. To model the balance constraints note that $\lfloor n/k \rfloor \leq |C_j| \leq \lceil n/k \rceil$ and that a coloring uses k colors if and only if $w_k - w_{k+1} = 1$, where w_k is an indicator variable that is 1 if color k is used and 0 otherwise.

2. Solve the model at point (1) in COMET with Solver MIP. Call $\chi_=(G)$ the minimum number of colors for which a coloring satisfying the balance condition is found. Report in a table like the one below the results for the 5 instances (one per each density). Keep the size fixed to the largest you could solve to optimality for all 5 different densities with the MIP model within a time limit of 60 seconds. (Use `m.displayModel()` to check the number of variables and constraints.)

$\|V\|$	$\|E\|$	ρ	Variables	Constraints	$\chi_=(G)$	Time
		0.1				
		0.3				
		0.5				
		0.7				
		0.9				

(Note that $\chi_=(G)$ is not necessarily the same as the chromatic number of a graph. For example, the star $K_{1,5}$ is a complete bipartite graph, and therefore may be colored with two colors while $\chi_=(G)$ is four.)

3. Formulate the problem for a fixed k as a Constraint Programming problem using a polynomial number of variables and constraints.

4. Implement the model in Comet inside a routine that minimizes k by iteratively solving CP models with decreasing k and halting when no solution can be found. Report results on the largest graphs you could consitently solve over ρ in a table

$\|V\|$	$\|E\|$	ρ	Variables	Constraints	Nodes	Propagations	Failures	Smallest k	Time
		0.1							
		0.3							
		0.5							
		0.7							
		0.9							

Note: focus your attention on the best formulation of the constraints while using always the default depth first search (the Assignment number 2 will have focus on the search aspect.)

5. Introduce `alldifferent` constraints to the previous CP formulation. To do this you need to find cliques in the graph. Write the Integer Linear Program that finds maximum cliques (hint: find independent sets in the complement graph). Further, implement a procedure that for each edge finds the maximal[1] clique that contains that edge. Call this set of cliques $\mathcal{C}$.

   Does the solution of the CP model improve with the introduction of these constraints? Report a table on the same graphs used for the CP model above.

6. The cliques in $\mathcal{C}$ can be used also in the MIP formulation. Write the constraints that you should add to the model at point (1).

7. Note that a graph may admit a k-coloring with balanced color classes but not one with $k+1$ colors. For example, the complete bipartite graph $K_{3,3}$ has a balanced 2-coloring but it does not have a balanced three coloring: it would have to have exactly two vertices per class, but the two sides of the bipartition cannot each be partitioned into pairs because they have an odd number of vertices. This result can be generalized to all bipartite graphs of the form $K_{2n+1,2n+1}$.

   Hence the iterative procedure at the previous point may not find $\chi_=(G)$. Devise an alternative procedure that using CP would find $\chi_=(G)$.

[1] The term maximum is used to denote the largest clique in the graph. The term maximal to denote a clique that cannot be extended any further. Clearly, a maximum clique is maximal but a maximal clique is not necessarily the maximum clique.

Describe the formulations and experiments requested in the points above in a short report (max 3 pages). Do not describe the problem. Submit the report together with the `.co` scripts. Organize the scripts as follows: `ip.co` containing the implementation required at point 2 and `cp.co` containing all other implementations required. Do not submit the instances or the scripts made available. Submit in the BlackBoard system. Use only your CPR number (all 10 digits without dash) or your exam number as identifier of the documents.
There is not a predefined score for each point but points 1 and 3 are the most important as from their correctness depends most of the remaining work. The quality of the models, determined by the size of the largest graphs solvable, is relevant for the final grade. Experimenting and inspecting solutions on small graphs may help to detect errors and find improvements.

2 Practicalities

This assignment is best carried out in COMET. All scripts mentioned below can be found at `http://www.imada.sdu.dk/~marco/DM826/Resources/Assignment1`

- For the computational experiments use uniform random graphs of different size and edge density $\rho \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$.

  To generate these graphs use the script `graph_generator.co`

  ```
  comet graph_generator.co -n25 -p0.5 -fg.dat
  ```

  The call generates a graph with 25 vertices and edge density 0.5 and writes it in DIMACS format in `g.dat`. Be aware that the graph may contain isolated vertices.

- For loading graphs in your script include the script `loadDIMACS`

  ```
  include "loadDIMACS";
  ```

  The file included will parse the file passed to your script when calling comet, ie,

  ```
  comet equi.co -fg.dat
  ```

  After execution there will be two data structures storing the graph, an adjacency matrix and a list of edges

  ```
  int nv; %number of vertices
  int me; % number of edges
  bool [,]Adj;
  tuple edge{int u; int v;}
  set{edge} Edges();
  ```

- Always impose in your experiments a total time limit of 60 seconds. Use:

  ```
  int t0 = System.getCPUTime();
  m.limitTime(60);
  cout << "Time: " << (System.getCPUTime() - t0)/1000 << " seconds" << endl;
  ```

  where m is a solver. You will need to use the solvers MIP, LP, CP, importing `cotfd` and `cotln`. The MIP module calls SCIP. For debugging purposes it may be useful

to inspect the output of SCIP. This can be done by adding `m.displayModel()`. The LP module uses COINOR CLP 1.8.2.

In the CP module, to count the number of nodes in the search tree and the number of failures use:

```
cout << "#choices = " << m.getNChoice() << endl;
cout << "#fail = " << m.getNFail() << endl;
cout << ‘‘#propagations = ‘‘<<m.getNPropag()<<endl;
```

To test if a solution has been found: `m.getSolution() != null`.