Combinatorial Problem Solving (CPS)

2017-18 Spring Term. Final Exam: 3 hours
Publication of grades: 28/06
Revision of exams: 02/07 (office 113, Ω building, under request by e-mail: `erodri@cs.upc.edu`)

1. *(4 pts.)* Let us consider variables $x_1, x_2, \ldots, x_n$ with respective domains $D_1, D_2, \ldots, D_n$ such that:

   i. $D_i \subseteq \{1, 2, \ldots, n\} - \{i\}$ for each $1 \leq i \leq n$.
   ii. $j \in D_i$ if and only if $i \in D_j$.

   The *symmetric alldifferent* constraint $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$ is defined as follows. A tuple $(v_1, v_2, \ldots, v_n)$ of values for $(x_1, x_2, \ldots, x_n)$ satisfies $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$ if and only if

   (1) $v_i \in D_i$ for all $1 \leq i \leq n$, and
   (2) $v_i \neq v_j$ for all $1 \leq i < j \leq n$, and
   (3) $v_i = j$ if and only $v_j = i$ for all $1 \leq i < j \leq n$.

   (a) *(1 pt.)* Consider the following problem. You are an activity leader in charge of 8 children. You have hired 4 paddle courts for a short time, so that the 8 children have to be grouped into 4 pairs to play a single match simultaneously, one pair for each of the courts.

   Moreover, the following compatibility relations have to be respected (to simplify the notation, we will identify the children with numbers from 1 to 8):

   - Child 1 can play with children 2 and 5.
   - Child 2 can play with children 1 and 5.
   - Child 3 can play with children 4 and 6.
   - Child 4 can play with children 3 and 6.
   - Child 5 can play with children 1, 2, 7 and 8.
   - Child 6 can play with children 3, 4, 7 and 8.
   - Child 7 can play with children 5, 6 and 8.
   - Child 8 can play with children 5, 6 and 7.

   Model this problem as a CSP that uses one $\mathsf{SymmAllDiff}$ constraint. Specify the variables, their meanings, their domains, and the constraints of the CSP.

   Solution:

   For each $1 \leq i \leq 8$ let us consider a variable x_i meaning "the child paired with child i". Taking into account the compatibility relations, the domains are:

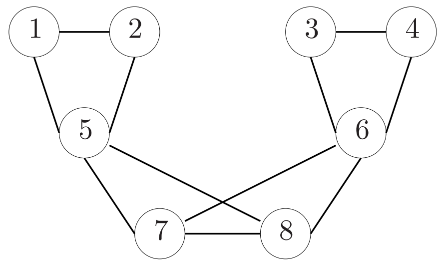
   - $D_1 = \{2, 5\}$
   - $D_2 = \{1, 5\}$
   - $D_3 = \{4, 6\}$
   - $D_4 = \{3, 6\}$
   - $D_5 = \{1, 2, 7, 8\}$
   - $D_6 = \{3, 4, 7, 8\}$
   - $D_7 = \{5, 6, 8\}$
   - $D_8 = \{5, 6, 7\}$

   The CSP has a single constraint $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_8)$.

(b) *(1 pt.)* The *value graph* of a constraint $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$ is the graph $G = (V, E)$ where $V = \{1, 2, \ldots, n\}$ and $E = \{\, \{i, j\} \mid i \in D_j,\ j \in D_i \,\}$.

Draw the value graph that corresponds to the SymmAllDiff constraint of exercise (a).

Solution:



(c) *(1 pt.)* Prove that there is a bijection between solutions to $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$ and matchings of the value graph that cover $\{1, 2, \ldots, n\}$.

Solution:

Let $G = (V, E)$ be the value graph of the constraint $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$. Given a solution $(v_1, v_2, \ldots, v_n)$ to $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$, we define $M = \{\{i, v_i\} \mid 1 \leq i \leq n\}$.

First of all we have that $M \subseteq E$: given $1 \leq i \leq n$, let $j = v_i$. Then $i = v_j$. Therefore $j = v_i \in D_i$ and $i = v_j \in D_j$. By definition, $\{i, j\} \in E$.
Moreover, M is a matching. Let us reason by contradiction. Let us assume there exist two different edges in M that coincide at a vertex. They will be of the form $\{i, j\}$ and $\{i, k\}$. But then $j = v_i = k$, which is a contradiction.
Finally, since for each $1 \leq i \leq n$ we have an edge $\{i, v_i\} \in M$, we conclude that M covers $\{1, 2, \ldots, n\}$.

The inverse mapping is as follows. Let us assume that $M \subseteq E$ is a matching of the value graph that covers $\{1, 2, \ldots, n\}$. For each edge $\{i, j\} \in M$, we define $v_i = j$.

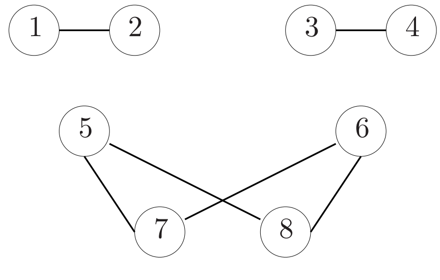
We claim that $(v_1, v_2, \ldots, v_n)$ is a solution to $\mathsf{SymmAllDiff}(x_1, x_2, \ldots, x_n)$. The tuple is well defined: as M is a matching that covers $\{1, 2, \ldots, n\}$, there is exactly one edge in M that is incident to i. Moreover:

$$v_i = j \Leftrightarrow \{i, j\} \in M \Leftrightarrow \{j, i\} \in M \Leftrightarrow v_j = i$$

And finally $i \neq j$ implies $v_i \neq v_j$: if $k = v_i = v_j$ then $i = v_k = j$.

(d) *(1 pt.)* Draw the value graph that corresponds to the SymmAllDiff constraint of exercise (a) after enforcing arc consistency.

Solution:



2. *(3 pts.)* Consider a linear program of the following form:

$$\begin{aligned} &\min c^T x \\ &Ax = b \\ &0 \leq x_i \leq 1 \quad \text{for all } 1 \leq i \leq n, \end{aligned}$$

where $c \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}, n \geq m$ and $\operatorname{rank}(A) = m$.

(a) *(1.5 pt.)* Let us assume we apply the bounded version of the simplex algorithm to solve the above linear program. Let us consider the basic solution determined by the basic variables $\mathcal{B}$, the non-basic variables $\mathcal{L}$ assigned to the lower bound, and the non-basic variables $\mathcal{U}$ assigned to the upper bound. Note that $\mathcal{B} \cup \mathcal{L} \cup \mathcal{U} = \{x_1, x_2, \ldots, x_n\}$.

Give the (simplest) formula that expresses the values of the basic variables in the basic solution in terms of the non-basic variables.

Solution:

Let B be the basis corresponding to the basic variables $\mathcal{B}$. Then the values $x_{\mathcal{B}}$ of the basic variables in the basic solution are:

$$x_{\mathcal{B}} = B^{-1}b - \sum_{x_j \in \mathcal{U}} B^{-1}a_j$$

(b) *(1.5 pts.)* Let us assume that $b \in \mathbb{Z}^m$ and $A \in \mathbb{Z}^{m \times n}$. Moreover, let us assume that A is *totally unimodular*: the determinant of every square submatrix of A is 0 or ± 1. Prove that, under these assumptions, if the above linear program has a finite optimum, then there exists an optimal solution $x^* = (x_1^*, x_2^*, \ldots, x_n^*)$ satisfying that $x_i^* \in \mathbb{Z}$ for all $1 \leq i \leq n$.

Hint: You can use Cramer's rule, which states that the inverse of an invertible matrix M is

$$M^{-1} = \frac{1}{\det(M)} C^T$$

where for all $1 \leq i, j \leq n$ the coefficient at row i and column j of C, the so-called cofactor matrix, *is $C_{ij} = (-1)^{i+j} D_{ij}$, and D_{ij} is the determinant of the $(n-1) \times (n-1)$ matrix that results from deleting row i and column j from M.*

Solution:

By hypothesis, the above linear program has a finite optimum. By applying the bounded version of the simplex algorithm we obtain an optimal solution determined by basic variables $\mathcal{B}$, non-basic variables $\mathcal{L}$ assigned to the lower bound, and non-basic variables $\mathcal{U}$ assigned to the upper bound.

In this basic optimal solution the assigned value to a non-basic variable is either its lower bound (which is 0) or its upper bound (which is 1). In any case, it is an integer value.

So it just remains to check the integrality condition for the basic variables. First of all we notice that B^{-1} is an integer matrix. Indeed, $A \in \mathbb{Z}^{m \times n}$ implies that $B \in \mathbb{Z}^{m \times m}$. Therefore the cofactor matrix of B is also integer, as it is obtained by computing determinants of matrices with integer coefficients. Finally, $\det(B) = \pm 1$ since B is an invertible submatrix of A, which is totally unimodular. By applying Cramer's rule we can deduce that B^{-1} is integer.

Moreover, by hypothesis $b \in \mathbb{Z}^m$, and $A \in \mathbb{Z}^{m \times n}$ implies that $a_j \in \mathbb{Z}^m$ for all $j \in \mathcal{U}$. Finally, using that

$$x_{\mathcal{B}} = B^{-1}b - \sum_{j \in \mathcal{U}} B^{-1}a_j$$

and that B^{-1} is an integer matrix and b and the a_j are integer vectors, we conclude that $x_{\mathcal{B}}$ is an integer optimal solution.

3. *(3 pts.)* We define the problem **NEG-SAT** as follows: given a propositional formula F, to determine whether there exists I such that $I \models \neg F$.

   (a) *(1.5 pts.)* Describe a linear-time algorithm for **NEG-SAT** when the input formula is in CNF. Justify its correctness and its cost.

   Hint: you can use that, given a clause C, detecting if C contains contradictory literals, i.e., p and $\neg p$ for some variable p, can be done in linear time.

   Solution:

   The algorithm works as follows. Let F be the input CNF. If F is the empty set of clauses, then F is a tautology, and hence we return NO. Otherwise we take a clause C of F. If there is a pair of contradictory literals in C, then C is a tautology that can be discarded, and the algorithm starts again with the resulting formula. Otherwise we can build an interpretation I that falsifies C by setting to false all the literals in C. This interpretation also falsifies F, and so we return YES.

   The algorithm is correct because when we return YES, we have indeed found an interpretation that falsifies F. For the reverse implication, if there exists I such that $I \models \neg F$, then there exists $C \in F$ such that $I \models \neg C$. The algorithm will return YES when this clause is processed.

   Since detecting contradictory literals can be done in linear time and we just need to make one pass over the formula, the algorithm is linear.

   (b) *(1.5 pts.)* Let us call CNF-NEG-SAT the linear-time algorithm of the previous exercise for **NEG-SAT** when the input formula is in CNF :

   Algorithm CNF-NEG-SAT

   Input: propositional formula F in CNF

   Output: YES if there exists I such that $I \models \neg F$, NO otherwise

   Consider now the following algorithm for solving the **SAT** problem for arbitrary formulas:

   Algorithm MY-SAT

   Input: propositional formula F

   Output: YES if there exists I such that $I \models F$, NO otherwise

   Step 1. $G :=$ Tseitin-transformation-of($\neg F$)
   Step 2. **return** CNF-NEG-SAT(G)

   Is algorithm MY-SAT correct? If so, prove it. Otherwise, give a counterexample.

   Solution:

   Algorithm MY-SAT is not correct. First of all notice that the Tseitin transformation G of any formula is of the form $p \wedge G'$, where G' is some CNF and p is the auxiliary propositional variable representing the root of the formula tree. So for any interpretation I such that $I(p) = 0$ we will have $I \not\models G$, i.e., $I \models \neg G$. So in algorithm MY-SAT(F), the call CNF-NEG-SAT(G) always returns YES, independently of F. In particular, even if F is unsatisfiable (for example, $q \wedge \neg q$), MY-SAT(F) will return YES.