Combinatorial Problem Solving (CPS)

2018-19 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. *(3 pts.)* Let $x_1, \ldots, x_n$ be integer variables. For each $1 \le i \le n$, the domain of variable x_i is the interval $[\ell_i, u_i]$, where $\ell_i, u_i \in \mathbb{Z}$ and $\ell_i \le u_i$.

   Let C be a constraint of the form $a_1 x_1 + \cdots + a_n x_n \ge k$, where $a_i, k \in \mathbb{Z}$ and $a_i > 0$.

   Let us consider the CSP consisting of the single constraint C.

   (a) *(1 pt.)* Prove that, if x is a solution to the CSP, then for each $1 \le i \le n$ we have that

   $$x_i \ge \left\lceil \frac{k - \sum_{j=1, j \ne i}^{n} a_j u_j}{a_i} \right\rceil$$

   Solution:

   If x is a solution to the CSP, then

   $$a_i x_i + \sum_{j=1, j \ne i}^{n} a_j u_j \ge a_i x_i + \sum_{j=1, j \ne i}^{n} a_j x_j = \sum_{j=1}^{n} a_j x_j \ge k.$$

   Therefore

   $$a_i x_i \ge k - \sum_{j=1, j \ne i}^{n} a_j u_j$$

   and since $a_i > 0$

   $$x_i \ge \frac{k - \sum_{j=1, j \ne i}^{n} a_j u_j}{a_i}$$

   Finally, as $x_i \in \mathbb{Z}$, we have that

   $$x_i \ge \left\lceil \frac{k - \sum_{j=1, j \ne i}^{n} a_j u_j}{a_i} \right\rceil$$

   (b) *(1 pt.)* Let $S = \sum_{j=1}^{n} a_j u_j$. Prove that, if $S - \max_{1 \le j \le n}(a_j(u_j - l_j)) < k$, then the CSP is arc-inconsistent.

   Solution:

   Let i with $1 \le i \le n$ be such that $a_i(u_i - l_i) = \max_{1 \le j \le n}(a_j(u_j - l_j))$. Then

   $$k > -a_i(u_i - l_i) + S = -a_i u_i + a_i l_i + \sum_{j=1}^{n} a_j u_j = a_i l_i + \sum_{j=1, j \ne i}^{n} a_j u_j$$

   So

   $$k - \sum_{j=1, j \ne i}^{n} a_j u_j > a_i l_i$$

   and as $a_i > 0$

   $$\left\lceil \frac{k - \sum_{j=1, j \ne i}^{n} a_j u_j}{a_i} \right\rceil \ge \frac{k - \sum_{j=1, j \ne i}^{n} a_j u_j}{a_i} > l_i$$

   Finally, by the previous exercise, we conclude that value l_i for variable x_i is arc-inconsistent.

(c) *(1 pt.)* Is the reverse implication of exercise (**??**) true? If so, prove it. Otherwise, give a counterexample.

Solution:

It is true. Let x_i be a variable with an arc-inconsistent value in its domain. Then value ℓ_i is arc-inconsistent, since if a value α for x_i has a support, then any value $\beta \geq \alpha$ also has a support.
Hence, if value ℓ_i for x_i does not have any support, in particular

$$a_i l_i + \sum_{j=1, j \neq i}^{n} a_j u_j < k.$$

So

$$S - a_i(u_i - l_i) = a_i l_i - a_i u_i + \sum_{j=1}^{n} a_j u_j = a_i l_i + \sum_{j=1, j \neq i}^{n} a_j u_j < k.$$

But since $a_i(u_i - l_i) \leq \max_{1 \leq j \leq n}(a_j(u_j - l_j))$, we conclude that

$$S - \max_{1 \leq j \leq n}(a_j(u_j - l_j)) \leq S - a_i(u_i - l_i) < k.$$

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

$$\begin{array}{l} \min c^T x \\ Ax = b \\ x \geq 0 \quad x_i \in \mathbb{Z} \ \text{ for all } 1 \leq i \leq n. \end{array}$$

Let β be a basic solution and

$$x_i = \gamma - \sum_{j \in R} a_j x_j$$

be the equation in the tableau of a basic variable x_i, where R are the indices of the non-basic variables and $\gamma, a_j \in \mathbb{R}$. Let us assume that $\beta_i \notin \mathbb{Z}$, where β_i is the value assigned by β to x_i.

(a) *(1.5 pts.)* Prove that

$$x_i + \sum_{j \in R} \lfloor a_j \rfloor x_j - \lfloor \gamma \rfloor = \gamma - \lfloor \gamma \rfloor - \sum_{j \in R} (a_j - \lfloor a_j \rfloor) x_j$$

for all feasible solutions to the integer linear program.

Solution:

$$\begin{array}{rcll} x_i + \sum_{j \in R} \lfloor a_j \rfloor x_j - \lfloor \gamma \rfloor & = & \gamma - \lfloor \gamma \rfloor - \sum_{j \in R}(a_j - \lfloor a_j \rfloor) x_j & \text{iff} \\ x_i + \sum_{j \in R} \lfloor a_j \rfloor x_j & = & \gamma - \sum_{j \in R}(a_j - \lfloor a_j \rfloor) x_j & \text{iff} \\ x_i + \sum_{j \in R} \lfloor a_j \rfloor x_j & = & \gamma - \sum_{j \in R} a_j x_j + \sum_{j \in R} \lfloor a_j \rfloor x_j & \text{iff} \\ x_i & = & \gamma - \sum_{j \in R} a_j x_j & \end{array}$$

(b) *(1.5 pts.)* Prove that

$$\gamma - \lfloor \gamma \rfloor - \sum_{j \in R} (a_j - \lfloor a_j \rfloor) x_j \leq 0$$

is a cut that cuts β away.

Solution:

First, let us prove that the inequality does not hold for β. Since β is a basic solution, we have that $\beta_j = 0$ for all $j \in R$. Hence, by substituting in

$$x_i = \gamma - \sum_{j \in R} a_j x_j,$$

we get that $\beta_i = \gamma$. Thus the inequality reduces to $\beta_i - \lfloor \beta_i \rfloor \leq 0$, which is false as $\beta_i \notin \mathbb{Z}$.

Now let us see that the inequality holds for all feasible solutions to the integer linear program. Let x be a feasible solution. Then in the equation

$$x_i + \sum_{j \in R} \lfloor a_j \rfloor x_j - \lfloor \gamma \rfloor = \gamma - \lfloor \gamma \rfloor - \sum_{j \in R} (a_j - \lfloor a_j \rfloor) x_j$$

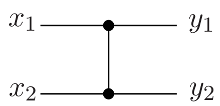
the left-hand side is an integer, and the right-hand size is a value < 1: indeed, $\gamma - \lfloor \gamma \rfloor < 1$, and $\sum_{j \in R} (a_j - \lfloor a_j \rfloor) x_j \geq 0$ as $a_j \geq \lfloor a_j \rfloor$ and $x_j \geq 0$. Since

$$\gamma - \lfloor \gamma \rfloor - \sum_{j \in R} (a_j - \lfloor a_j \rfloor) x_j$$

is an integer < 1, it must be ≤ 0.

3. *(4 pts.)* Answer the following questions:

(a) *(1.5 pts.)* Using the standard representation of a 2-comparator with inputs x_1, x_2 and outputs y_1, y_2 (in decreasing order):

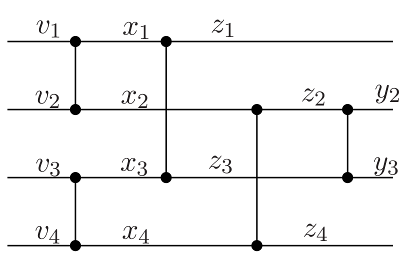


draw the circuit corresponding to a sorting network with 4 inputs v_1, v_2, v_3, v_4. Please indicate **clearly** the names of the auxiliary variables representing the wires of the circuit. Write also the set of clauses corresponding to the circuit.

Note: You can write the clauses as disjunctions or as implications.

Solution:

The circuit:



The set of clauses:

$$
\begin{array}{lllllll}
x_1 & \rightarrow & v_1 \vee v_2 & \qquad & x_2 & \rightarrow & v_1 \\
x_1 & \leftarrow & v_1 & & x_2 & \rightarrow & v_2 \\
x_1 & \leftarrow & v_2 & & x_2 & \leftarrow & v_1 \wedge v_2 \\
\\
x_3 & \rightarrow & v_3 \vee v_4 & & x_4 & \rightarrow & v_3 \\
x_3 & \leftarrow & v_3 & & x_4 & \rightarrow & v_4 \\
x_3 & \leftarrow & v_4 & & x_4 & \leftarrow & v_3 \wedge v_4 \\
\\
z_1 & \rightarrow & x_1 \vee x_3 & & z_3 & \rightarrow & x_1 \\
z_1 & \leftarrow & x_1 & & z_3 & \rightarrow & x_3 \\
z_1 & \leftarrow & x_3 & & z_3 & \leftarrow & x_1 \wedge x_3 \\
\\
z_2 & \rightarrow & x_2 \vee x_4 & & z_4 & \rightarrow & x_2 \\
z_2 & \leftarrow & x_2 & & z_4 & \rightarrow & x_4 \\
z_2 & \leftarrow & x_4 & & z_4 & \leftarrow & x_2 \wedge x_4 \\
\\
y_2 & \rightarrow & z_2 \vee z_3 & & y_3 & \rightarrow & z_2 \\
y_2 & \leftarrow & z_2 & & y_3 & \rightarrow & z_3 \\
y_2 & \leftarrow & z_3 & & y_3 & \leftarrow & z_2 \wedge z_3
\end{array}
$$

(b) *(1.5 pts.)* A *pseudo-boolean constraint* is a constraint of the form $a_1x_1 + \cdots + a_mx_m \leq k$, where $k, a_1, \ldots, a_m$ are positive integers and $x_1, \ldots, x_m$ are boolean variables. Explain how to encode into SAT a pseudo-boolean constraint using sorting networks. Illustrate your method with the constraint $v_1 + 3v_2 \leq 2$ and give the resulting set of clauses.

Solution:

We notice that $a_1x_1 + \cdots + a_mx_m = \overbrace{x_1 + \cdots + x_1}^{a_1} + \overbrace{x_2 + \cdots + x_2}^{a_2} + \cdots + \overbrace{x_m + \cdots + x_m}^{a_m}$. Let $n = \sum_{i=1}^m a_i$. We consider a sorting network of size n in which the first a_1 inputs are (clones of) the variable x_1, the following a_2 inputs are the variable x_2, etc. The encoding consists of the clauses of the circuit and the unit clause $\overline{y}_{k+1}$, where y_{k+1} is the $(k+1)$-th output of the sorting network (if $k < n$; otherwise, the constraint is trivially true).

For the constraint $v_1 + 3v_2 \leq 2$, the clauses would be $\overline{y_3}$ and:

$$
\begin{array}{lllllll}
x_1 & \rightarrow & v_1 \vee v_2 & \qquad & x_2 & \rightarrow & v_1 \\
x_1 & \leftarrow & v_1 & & x_2 & \rightarrow & v_2 \\
x_1 & \leftarrow & v_2 & & x_2 & \leftarrow & v_1 \wedge v_2 \\
\\
x_3 & \rightarrow & v_2 & & x_4 & \rightarrow & v_2 \\
x_3 & \leftarrow & v_2 & & x_4 & \leftarrow & v_2 \\
\\
z_1 & \rightarrow & x_1 \vee x_3 & & z_3 & \rightarrow & x_1 \\
z_1 & \leftarrow & x_1 & & z_3 & \rightarrow & x_3 \\
z_1 & \leftarrow & x_3 & & z_3 & \leftarrow & x_1 \wedge x_3 \\
\\
z_2 & \rightarrow & x_2 \vee x_4 & & z_4 & \rightarrow & x_2 \\
z_2 & \leftarrow & x_2 & & z_4 & \rightarrow & x_4 \\
z_2 & \leftarrow & x_4 & & z_4 & \leftarrow & x_2 \wedge x_4 \\
\\
y_2 & \rightarrow & z_2 \vee z_3 & & y_3 & \rightarrow & z_2 \\
y_2 & \leftarrow & z_2 & & y_3 & \rightarrow & z_3 \\
y_2 & \leftarrow & z_3 & & y_3 & \leftarrow & z_2 \wedge z_3
\end{array}
$$

(c) *(1 pt.)* Is your encoding of exercise (**??**) arc-consistent? (that is, if a value of a variable does not have a support for the constraint, does unit propagation in the CNF propagate a literal that discards that value?)

If it is so, prove it. Otherwise, give a counterexample.

Solution:

The encoding is not arc-consistent. For example, let us consider the constraint of the previous exercise and its codification into SAT. If the encoding were arc-consistent, unit propagation should propagate $\overline{v_2}$, since v_2 cannot be true. But we notice that the only literal that can be propagated is $\overline{y_3}$.