ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institute of Theoretical Computer Science
Bernd Gärtner, Mohsen Ghaffari, Rasmus Kyng, David Steurer



## Solution 1

(i) $\implies$ (iii) Let $\bar{A}x \leq \bar{b}$ denote the subsystem of all $y$-tight constraints, where we mean by a $y$-tight constraint a constraint that $y$ fulfills with equality. Now, we define $c := \bar{A}^T\mathbf{1}$, where $\mathbf{1}$ denotes the all-ones vector in $k$ dimensions, with $k$ being the number of $y$-tight constraints. We have

$$c^T y = \mathbf{1}^T \bar{A} y = \mathbf{1}^T \bar{b}.$$

Now, let $z \in P \setminus \{y\}$ be arbitrary. As $y$ is a basic feasible solution, it satisfies $n$ linearly independent constraints with equality and hence the rank of $\bar{A}$ is $n$. In particular, this implies that $z$ cannot satisfy all of the $y$-tight constraints with equality. Hence, we get

$$c^T z = \mathbf{1}^T \bar{A} z < \mathbf{1}^T \bar{b}$$

and therefore

$$c^T y > c^T z,$$

as desired.

(iii) $\implies$ (ii) Let $c \in \mathbb{R}^n$ such that for every $z \in P \setminus \{y\}$, $c^T y > c^T z$. Now, for the sake of contradiction, assume that there exist two distinct points $y_1, y_2 \in P$ with $y = \frac{1}{2}(y_1 + y_2)$. This implies

$$c^T y = c^T \left(\frac{1}{2}(y_1 + y_2)\right) = \frac{1}{2}\left(c^T y_1 + c^T y_2\right) < \frac{1}{2}\left(c^T y + c^T y\right) = c^T y.$$

This is a contradiction and therefore there do not exist two distinct points $y_1, y_2 \in P$ with $y = \frac{1}{2}(y_1 + y_2)$.

(ii) $\implies$ (i)

We show the contrapositive. Assume that $y$ is not a basic feasible solution. Let $\bar{A}x \leq \bar{b}$ denote the subsystem of all $y$-tight constraints. As $y$ is not a basic feasible solution, the rank of $\bar{A}$ is strictly less than $n$ and hence there exists a non-zero vector $v \in \mathbb{R}^n$ with $\bar{A}v = \mathbf{0}$, where $\mathbf{0}$ denotes the all-zero vector. For $\varepsilon > 0$, by defining $y_\varepsilon := y + \varepsilon v$ and $y_{-\varepsilon} := y - \varepsilon v$, we get $y_\varepsilon \neq y_{-\varepsilon}$ and $y = \frac{1}{2}(y_\varepsilon + y_{-\varepsilon})$. Thus, it remains to show that for small enough $\varepsilon$, both $y_\varepsilon$ and $y_{-\varepsilon}$ are contained in $P$. Note that for the $y$-tight constraints, we have

$$\bar{A}y_\varepsilon = \bar{A}y + \varepsilon\bar{A}v = \bar{b} + \mathbf{0} = \bar{b}$$

and similar for $y_{-\varepsilon}$. Moreover, by choosing $\varepsilon$ small enough, we can also ensure that $y_\varepsilon$ and $y_{-\varepsilon}$ satisfy all the non $y$-tight constraints. Hence, we have shown the contrapositive.

## Solution 2

(a) Consider the following Integer Linear Program

$$\begin{array}{lrl} \text{minimize} & \sum_{v\in V} y_v & \\ \text{subject to} & y_u + y_v \geq 1 & \forall\, \{u,v\} \in E \\ & y_v \in \{0,1\} & \forall\, v \in V. \end{array} \tag{1}$$

Let $y \in \{0,1\}^V$ denote an arbitrary feasible solution to the ILP. Now, let $V_{cover} := \{v \in V\colon y_v = 1\}$. To prove that $V_{cover}$ is a valid vertex cover, consider an arbitrary edge $\{u,v\} \in E$. As $y_u + y_v \geq 1$, we either have $y_v = 1$, which implies that $v \in V_{cover}$, or if $y_v = 0$, then $y_u = 1$ and therefore $u \in V_{cover}$. In both cases at least one of the endpoints of the edge $\{u,v\}$ is in $V_{cover}$ and therefore $V_{cover}$ is a valid vertex cover. Moreover, we have $\sum_{v\in V} y_v = |V_{cover}|$. Now, let $V_{cover}$ denote an arbitrary vertex cover. Let $y \in \{0,1\}^V$ with $y_v = 1$ if $v \in V_{cover}$ and $y_v = 0$ otherwise. For every edge $\{u,v\} \in E$ at least one of the endpoints is contained in $V_{cover}$ and therefore $y_u + y_v \geq 1$. Hence, $y$ is a feasible solution to the ILP with $\sum_{v\in V} y_v = |V_{cover}|$.

(b) We relax the ILP as follows

$$\begin{array}{lrl} \text{minimize} & \sum_{v\in V} y_v & \\ \text{subject to} & y_u + y_v \geq 1 & \forall\, \{u,v\} \in E \\ & 0 \leq y_v \leq 1 & \forall\, v \in V. \end{array} \tag{2}$$

Let $y \in \mathbb{R}^V$ denote an arbitrary feasible solution to the LP. We now consider a procedure that runs in polynomial time with the following property: If $y$ is not an integral solution, then it finds a feasible solution $\tilde{y} \in \mathbb{R}^V$ with $\sum_{v\in V} \tilde{y}_v \leq \sum_{v\in V} y_v$. Moreover, the number of non-integral entries of $\tilde{y}$ is strictly smaller than the number of non-integral entries of $y$. Hence, by repeatedly applying this procedure for at most $n$ times, we arrive at a feasible integral solution $y_{int} \in \mathbf{R}^V$ with $\sum_{v\in V}(y_{int})_v \leq \sum_{v\in V} y_v$. As $y_{int}$ is a feasible solution to the ILP, we can convert it to a feasible vertex cover $V_{cover}$ with $|V_{cover}| \leq \sum_{v\in V}(y_{int})_v \leq \sum_{v\in V} y_v$.

The basic idea of the procedure is to increase the value of all fractional entries by $\varepsilon$ on one side of the bipartition and to decrease the value of all fractional entries by $\varepsilon$ on the other side of the bipartition. Let $V = U \sqcup W$ be an arbitrary bipartition of the vertices and let $y$ denote a fractional solution. Now, we define

$$U_{frac} := \{u \in U\colon y_u \in (0,1)\}$$

and

$$W_{frac} := \{w \in W\colon y_w \in (0,1)\}.$$

We assume without loss of generality that $|U_{frac}| \geq |W_{frac}|$. Let $\varepsilon_U := \min_{u \in U_{frac}} y_u$ , $\varepsilon_W := \min_{w \in W_{frac}} 1 - y_w$ and $\varepsilon := \min(\varepsilon_U, \varepsilon_W)$. We now consider $\tilde{y} \in \mathbb{R}^V$ with

$$\tilde{y}_v = \begin{cases} y_v & y_v \in \{0,1\} \\ y_v - \varepsilon & v \in U_{frac} \\ y_v + \varepsilon & v \in W_{frac}. \end{cases}$$

First, we verify that $\tilde{y}$ is a feasible solution. From the definition of $\varepsilon$ and a simple case distinction, it follows that $\tilde{y}_v \in [0,1]$ for every $v \in V$. Now, consider some arbitrary edge $\{u,w\} \in E$ with $u \in U$ and $w \in W$. If $y_u = 1$, then $\tilde{y}_u = 1$ and $y_u = 0$ implies $y_w = 1$ and therefore $\tilde{y}_w = 1$. In both cases $\tilde{y}_u + \tilde{y}_w \geq 1$. Thus, it remains to consider the case that $y_u \in (0,1)$. As $y_u + y_w \geq 1$, this implies $y_w > 0$. If $y_w = 1$, we again have $\tilde{y}_u + \tilde{y}_w \geq 1$ and if $y_w < 1$, then

$$\tilde{y}_u + \tilde{y}_w = y_u - \varepsilon + y_w + \varepsilon = y_u + y_w \geq 1.$$

Hence, $\tilde{y}$ is indeed a feasible solution. Moreover, we have

$$\sum_{v \in V} \tilde{y}_v = \sum_{v \in V} y_v + \varepsilon(|W_{frac}| - |U_{frac}|) \leq \sum_{v \in V} y_v,$$

where the last inequality follows from our assumption $|U_{frac}| \geq |W_{frac}|$. Thus, it remains to verify that the number of non-integral entries in $\tilde{y}$ is strictly smaller than the number of non-integral entries in $y$. Note that each integral entry in $y$ is also integral in $\tilde{y}$. Hence, it suffices to show that there exists at least one non-integral entry of $y$ that is integral in $\tilde{y}$. From the definition of $\varepsilon$, there either exists a $u \in U_{frac}$ with $\varepsilon = y_u$, in which case $\tilde{y}_u = 0$, or otherwise there exists a $w \in W_{frac}$ with $\varepsilon = 1 - y_w$, in which case $\tilde{y}_w = 1$. This finishes the proof and the overall procedure clearly runs in polynomial time.

(c) As the encoding size of the relaxed LP is polynomial in the encoding size of the input, we can use the Ellipsoid Method to find an optimal solution to the relaxed LP in polynomial (such an optimal solution clearly exists). We then use the polynomial time algorithm of the previous exercise to find a vertex cover whose size is at most the optimal value of the relaxed LP and hence a minimum vertex cover. The overall runtime is polynomial in the input size. It remains to verify that the procedure runs in polynomial time. Note that we haven't specified the underlying computational model. In case the algorithm is allowed to perform basic operations on real numbers, then our algorithm clearly runs in time polynomial in $n$. In case we are interested in the bit complexity, i.e., we would like to run the algorithm on a Turing Machine, then it is easy to verify that the algorithm runs in polynomial time in the input size.

## Solution 3

(a) First, consider the case that $T_1$ and $T_2$ are isomorphic and let $f$ be a function from the vertices of $T_1$ to the vertices of $T_2$ such that for each vertex $v$ of $T_1$ with children $v_1, v_2, \ldots, v_k$ the children of $f(v)$ in $T_2$ are exactly $f(v_1), f(v_2), \ldots, f(v_k)$. We prove the statement by induction on the height of the subtree rooted at $v$ that $P_v = P_{f(v)}$ for each vertex $v$ of $T$. The base case is that $v$ is a leaf. This implies that $f(v)$ is also a leaf. Furthermore, one can also easily establish by induction that the height of $v$ in $T_1$ is equal to the height of $f(v)$ in $T_2$ and therefore

$P_v = P_{f(v)}$. Now, consider some arbitrary non-leaf vertex $v$ with children $v_1, v_2, \ldots, v_k$. By the induction hypothesis, we have $P_{v_i} = P_{f(v_i)}$ for every $i \in [k]$ and therefore

$$P_v = (x_h - P_{v_1}) \cdot (x_h - P_{v_2}) \cdot \ldots \cdot (x_h - P_{v_k}) = (x_h - P_{f(v_1)}) \cdot (x_h - P_{f(v_2)}) \cdot \ldots \cdot (x_h - P_{f(v_k)}) = P_{f(v)},$$

where the last equality follows as $f(v_1), f(v_2), \ldots, f(v_k)$ are all of the children of $f(v)$ and $v$ and $f(v)$ have the same height, thus finishing the induction proof.

Next, consider the case that $T_1$ and $T_2$ are not isomorphic. We prove by induction on the height of the subtree rooted at $v$ that for every vertex $v$ of $T_1$ and every vertex $u$ of $T_2$, with both $u$ and $v$ having the same height in the tree, the subtree rooted at $v$ not being isomorphic to the subtree rooted at $u$ implies that $P_v \neq P_u$. First, consider the case that $v$ is a leaf. If $u$ is also a leaf, then $v$ and $u$ are isomorphic, hence it suffices to consider the case that $u$ is not a leaf. Let $u_1, u_2, \ldots, u_k$ denote the children of $u$. Let $h$ denote the height of $v$ in $T_1$ and of $u$ in $T_2$. Then, $P_v := x_h$ and $P_u := (x_h - P_{u_1}) \cdot (x_h - P_{u_2}) \cdot \ldots \cdot (x_h - P_{u_k})$. Note that it more or less directly follows from the definition that the polynomial associated with each vertex is not the zero polynomial (This is not so clear over a finite field, though). Hence, we can set the variables $x_{h+1}, x_{h+2}, \ldots$ in such a way that the polynomial $P_{u_1}$ evaluates to something non-zero. If we now set $x_h = P_{u_1}(x_{h+1}, x_{h+2}, \ldots)$, then $P_v$ evaluates to something non-zero, while $P_u$ evaluates to zero, hence finishing the base case. Next, consider the case that $v$ is not a leaf and has children $v_1, v_2, \ldots, v_k$. Now, let $u$ be some node in $T_2$ having the same height as $v$. If $u$ is a leaf, then we can use the same argument as before, with the roles of $u$ and $v$ being reversed, to argue that $P_v \neq P_u$. Hence, we can assume that $u$ is not a leaf and has children $u_1, u_2, \ldots, u_r$. Now, for each $i \in [k]$, let

$$N_{i,1} := |\{j \in [k] \colon \text{The subtree rooted at } v_i \text{ is isomorphic to the subtree rooted at } v_j\}|$$

and

$$N_{i,2} := |\{j \in [r] \colon \text{The subtree rooted at } v_i \text{ is isomorphic to the subtree rooted at } u_j\}|.$$

If the subtree rooted at $v$ is not isomorphic to the subtree rooted at $u$, then there exists an $i \in [k]$ with $N_{i,1} \neq N_{i,2}$. To simplify the exposition, we assume that $N_{i,1} > N_{i,2}$. Now, we choose $x_i$ for $i \geq h+1$ independently and uniformly at random from the set $\{1, 2, \ldots, 10n^2\}$. By Schwartz-Zippel, the induction hypothesis and the fact that all polynomials under consideration have a degree of at most $n$, it follows that $P_{v_i}(x_{h+1}, x_{h+2}, \ldots) = P_{u_j}(x_{h+1}, x_{h+2}, \ldots)$ with probability at most $\frac{n}{10n^2} = \frac{1}{10n}$ for every $j \in [r]$ such that the subtree rooted at $v_i$ is not isomorphic to the subtree rooted at $u_j$. By a Union Bound, it follows that with strictly positive probability $P_{v_i}(x_{h+1}, x_{h+2}, \ldots) \neq P_{u_j}(x_{h+1}, x_{h+2}, \ldots)$ for every $j \in [r]$ such that the subtree rooted at $v_i$ is not isomorphic to the subtree rooted at $u_j$. In that case, let $Q_v(x_h)$ and $Q_u(x_h)$ denote the univariate polynomials obtained from $P_v$ and $P_u$, respectively, by fixing the variables $x_{h+1}, x_{h+2}, \ldots$. Note that the previous discussion implies that there exists a real number $z$ such that the multiplicity of the root $z$ in $Q_v(x_h)$ is strictly larger than the multiplicity of the root $z$ in $Q_u(x_h)$. This implies that $Q_v(x_h) \neq Q_u(x_h)$. To see why, note that $Q_u$ tends faster to zero than $Q_v$ for values close to $z$. Hence, $P_v$ is not equal to $P_u$, finishing the induction step and completing the proof.

(b) By strong induction on the size of the subtree of $v$ one can show that for each node $v$, the degree of $P_v$ is at most equal to the number of nodes in the subtree rooted at $v$. Hence, the degree of both $P_{r_1}$ and $P_{r_2}$ is at most $n$. As $P_{r_1}$ and $P_{r_2}$ are not the same polynomial, $P_{r_1} - P_{r_2}$ is not the zero-polynomial and has degree at most $n$. Hence, the number of $(h+1)$-tuples in $S^{h+1}$ that are zeros of $P_{r_1} - P_{r_2}$ is at most $n \cdot |S|^h$ and therefore the probability that we chose such a tuple from $S$ is at most $n/|S| = O(1/n)$. Hence, if $P_{r_1}$ and $P_{r_2}$ are not the same polynomial, then $P_{r_1}(x) \neq P_{r_2}(x)$ with probability $1 - O(1/n)$.

(c) Let $n_v$ denote the number of nodes in the subtree rooted at $v$. We show by strong induction on the size of the subtree that $|P_v(x)| \leq n^{4n_v-2}$. First, consider the case that $v$ is a leaf. In that case, $|P_v(x)| \leq n^2 = n^{4n_v-2}$. Now, assume that $v$ has children $v_1, v_2, \ldots, v_k$. The induction hypothesis implies that

$$\begin{aligned}
|P_v| &= |(x_h - P_{v_1}) \cdot (x_h - P_{v_2}) \cdot \ldots \cdot (x_h - P_{v_k})| \\
&\leq |(\max(2, x_h) \cdot \max(2, |P_{v_1}|)) \cdot (\max(2, x_h) \cdot \max(2, |P_{v_2}|)) \cdot \ldots \cdot (\max(2, x_h) \cdot \max(2, |P_{v_k}|))| \\
&\leq \max(2, x_h)^k \cdot \max(2, |P_{v_1}|) \cdot \ldots \cdot \max(2, |P_{v_k}|) \\
&\leq n^{2k} \cdot n^{4n_{v_1}-2} \cdot \ldots \cdot n^{4n_{v_k}-2} \\
&= n^{4\sum_{i=1}^{k} n_{v_i}} \\
&\leq n^{4n_v-2},
\end{aligned}$$

and thus we have finished the induction. In particular, we have shown that $|P_{r_1}(x) - P_{r_2}(x)| \leq 2n^{4n}$. Note that $P_{r_1}(x) \mod p = P_{r_2}(x) \mod p$ if and only if $p$ divides $P_{r_1}(x) - P_{r_2}(x)$. There are at most $O(\log(2n^{4n})) = O(n \log n)$ distinct primes that divide $P_{r_1}(x) - P_{r_2}(x)$ and as $h(k) = \Omega(k/\log(k))$, we choose a given prime with probability at most $O(\log(n^3)/n^3)$. Hence, a Union Bound implies that $P_{r_1}(x) \mod p = P_{r_2}(x) \mod p$ with probability $O(n \log(n) \cdot \log^3(n)/n^3) = O(1/n)$, as desired.

(d) Our algorithm first chooses $x \in S^{h+1}$ uniformly at random and then independently a prime $p$ uniformly at random from the set of all primes in the range from 1 to $n^3$. If $P_{r_1}(x) \mod p = P_{r_2}(x) \mod p$, then our algorithm outputs "Yes" and otherwise our algorithm outputs "No". If $T_1$ and $T_2$ are isomorphic, then a) implies that our algorithm always outputs "Yes". If $T_1$ and $T_2$ are not isomorphic, then a), b) and c) together with a simple Union Bound over two events implies that our algorithm outputs "Yes" with probability $O(1/n)$. Hence, it remains to analyze the runtime. First, we discuss how to choose a prime $p$ uniformly at random from the set of all primes in the range from 1 to $n^3$. To that end, we repeatedly choose a number uniformly at random from 1 to $n^3$ and check in $O(\log^{100}(n))$ time whether the number is prime. If so, set $p$ to this number. If not, then we repeat the procedure (with fresh randomness). If we haven't found a prime after $O(\log^3 n)$ repetitions, then our algorithm simply outputs "Yes". Note that if we indeed output a prime, then every prime between 1 and $n^3$ has the same probability to be chosen. As in each iteration we have a probability of $\Omega(1/\log(n))$ to find a prime, a Chernoff Bound implies that we find a prime with probability $1 - e^{-\Omega(\log^2 n)}$. Hence, our algorithm still succeeds with probability $1 - O(1/n)$. Thus, it remains to discuss how we can efficiently evaluate $P_{r_1}(x) \mod p$ and $P_{r_2}(x) \mod p$. Let $v$ be a non-leaf vertex with children $v_1, v_2, \ldots, v_k$. As we have

$$\begin{aligned}
P_v(x) \mod p &= \big((x_h - P_{v_1}(x)) \cdot (x_h - P_{v_2}(x)) \cdot \ldots \cdot (x_h - P_{v_k}(x))\big) \mod p \\
&= \big((x_h - (P_{v_1}(x) \mod p)) \cdot (x_h - (P_{v_2}(x) \mod p)) \cdot \ldots \cdot (x_h - (P_{v_k}(x) \mod p))\big) \mod p,
\end{aligned}$$

given $P_{v_1}(x) \mod p$, $P_{v_2}(x) \mod p, \ldots, P_{v_k}(x) \mod p$, we can compute $P_v(x) \mod p$ in $O(k)$ time. Hence, there exists a constant $c \geq 10$ such that given $P_{v_1}(x) \mod p$, $P_{v_2}(x) \mod p, \ldots, P_{v_k}(x) \mod p$, we can compute $P_v(x) \mod p$ in $c \cdot k$ time. We now prove by strong induction on the size of the subtree of $v$ that we can evaluate $P_v(x)$ in $c \cdot (2n_v - 1)$ time. If $v$ is a leaf, we have $c \cdot (2n_v - 1) \geq 10$, so it clearly holds. Now, let $v$ be a vertex with children $v_1, v_2, \ldots, v_k$. By the induction hypothesis, we can evaluate $P_v(x) \mod p$ in time

$$c \cdot k + \sum_{i=1}^{k} c \cdot (2n_{v_i} - 1) \leq c(2n_v - 1),$$

hence finishing the induction. In particular, we can evaluate both $P_{r_1}(x) \mod p$ and $P_{r_2}(x) \mod p$ in $O(n)$ time, thus proving that our algorithm runs in time $O(n)$.