

Online Learning for Min-Max Discrete Problems

Theoretical Computer Science Vol. 930 (2022) 209–217.

E. Bampis, D. Christou, V. Escoffier, K. T. Nguyen

Speaker: Joseph Chuang-Chieh Lin

Department of Computer Science & Information Engineering,
Tamkang University

30 June 2023

Outline

- 1 Introduction
 - The Online Learning Framework
 - Main Contribution
- 2 Main Theorem I
 - The Proof
 - An OGD for Online Min-Max-VC
- 3 Main Theorem II
 - Multi-Instance Min-Max VC
 - Multi-Instance Min-Max Perfect Matching

Outline

- 1 Introduction
 - The Online Learning Framework
 - Main Contribution
- 2 Main Theorem I
 - The Proof
 - An OGD for Online Min-Max-VC
- 3 Main Theorem II
 - Multi-Instance Min-Max VC
 - Multi-Instance Min-Max Perfect Matching

Online learning framework (1/4)

We focus on cost minimization problems.

- Decision space: \mathcal{X} .
- State space: \mathcal{Y} .
- Cost function $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$.

Online learning framework (1/4)

We focus on cost minimization problems.

- Decision space: \mathcal{X} .
- State space: \mathcal{Y} .
- Cost function $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathbb{R}$.

A perspective of an iterative adversarial game with T rounds.

- 1 The algorithm first chooses an action $\mathbf{x}^t \in \mathcal{X}$.
- 2 The (adversarial) nature reveals $\mathbf{y}^t \in \mathcal{Y}$ that could depend on \mathbf{x}^t .
- 3 The algorithm observes the state \mathbf{y}^t and suffers a loss $f^t(\mathbf{x}^t) = f(\mathbf{x}^t, \mathbf{y}^t)$.
 - The objective after observing \mathbf{y}^t : $f^t(\mathbf{x})$.

Online learning framework (2/4)

The objective of the player: minimize the accumulative cost

$$\sum_{t=1}^T f(\mathbf{x}^t, \mathbf{y}^t).$$

Online Learning Algorithms

An algorithm that decides the actions \mathbf{x}^t before observing \mathbf{y}^t for each t .

- The efficiency measure:

$$R_T = \sum_{t=1}^T f(\mathbf{x}^t, \mathbf{y}^t) - \sum_{t=1}^T f(\mathbf{x}^*, \mathbf{y}^t),$$

where $\mathbf{x}^* = \arg \min_{\mathbf{x} \in \mathcal{X}} \sum_{t=1}^T f(\mathbf{x}, \mathbf{y}^t)$ (static).

Online learning framework (3/4)

- We aim for algorithms with $R_T = O(T^c)$, for $0 \leq c < 1$.
 - Vanishing regret (or no-regret).

Online learning framework (3/4)

- We aim for algorithms with $R_T = O(T^c)$, for $0 \leq c < 1$.
 - Vanishing regret (or no-regret).
- A computational efficiency concern:

Online learning framework (3/4)

- We aim for algorithms with $R_T = O(T^c)$, for $0 \leq c < 1$.
 - Vanishing regret (or no-regret).
- A computational efficiency concern:
 - It's NP-hard to compute \mathbf{x}_t 's even for $T = 1$ and \mathbf{y}^1 is revealed beforehand.

A relaxed notion: α -regret

$$R_T^\alpha = \sum_{t=1}^T f(\mathbf{x}^t, \mathbf{y}^t) - \alpha \sum_{t=1}^T f(\mathbf{x}^*, \mathbf{y}^t),$$

- Goal: vanishing α -regret for some $\alpha \geq 1$.

Online learning framework (4/4)

Polynomial Time Vanishing α -Regret Algorithms

An online learning algorithm which

- computes \mathbf{x}^t in $\text{poly}(n, t)$, where n is the input instance size.
 - the (expected) regret is bounded by $\text{poly}(n)T^c$, for some constant $0 \leq c < 1$.
- For the case $\alpha = 1$, we call it a **polynomial time vanishing regret algorithm**.

Online learning framework (4/4)

Polynomial Time Vanishing α -Regret Algorithms

An online learning algorithm which

- computes \mathbf{x}^t in $\text{poly}(n, t)$, where n is the input instance size.
 - the (expected) regret is bounded by $\text{poly}(n)T^c$, for some constant $0 \leq c < 1$.
- For the case $\alpha = 1$, we call it a polynomial time vanishing regret algorithm.

The regret is polynomial in n and sublinear in T .

Main Contribution (1/8)

Cardinality constrained problems

Given an n -elements set \mathcal{U} , a set of constraints \mathcal{C} on $2^{\mathcal{U}}$, and an integer k .

Goal: Determine whether there exists a feasible solution of size $\leq k$.

Min-Max- \mathcal{P}

Given a cardinality problem \mathcal{P} where all the elements in \mathcal{U} are given non-negative weights.

Goal: compute a feasible solution such that the maximum weight of all its elements is minimized.

Main Contribution (2/8)

Online Min-Max- \mathcal{P}

An online learning variant of min-max- \mathcal{P} such that

- the set of elements in \mathcal{U} and the set of constraints \mathcal{C} remain **static**.
- the **weights** on the elements of \mathcal{U} **change over time**.

Main Contribution (2/8)

Online Min-Max- \mathcal{P}

An online learning variant of min-max- \mathcal{P} such that

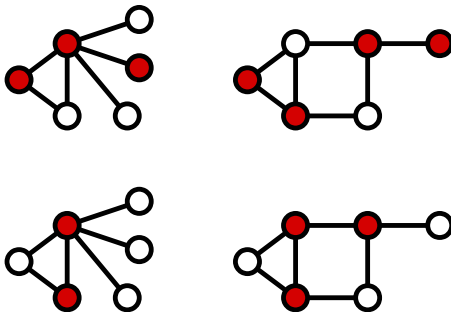
- the set of elements in \mathcal{U} and the set of constraints \mathcal{C} remain **static**.
- the **weights** on the elements of \mathcal{U} **change over time**.

Example: Min-Max Vertex Cover

- **Static:** Given a graph $G = (V, E)$, where each $v \in V$ has weight $w(v) \geq 0$. Find a vertex cover $V' \subseteq V$ which minimizes $w(V') = \max\{w(v) \mid v \in V'\}$.
- **Online-version:**
 - There are T rounds, a weight function w^t on the vertices for each round t .
 - An algorithm has to pick a vertex cover V'_t of G and suffers a loss $w(V'_t) = \max\{w(v) : v \in V'_t\}$.

Vertex Cover (VC)

Miym, CC BY-SA 3.0, via Wikimedia Commons



Static Min-Max VC is polynomial-time solvable

- VC_W : Given an integer W , determine if G has a vertex cover of maximum weight $\leq W$.
 - Pick all vertices of weight $\leq W$ and see if this is a vertex cover.

Static Min-Max VC is polynomial-time solvable

- VC_W : Given an integer W , determine if G has a vertex cover of maximum weight $\leq W$.
 - Pick all vertices of weight $\leq W$ and see if this is a vertex cover.
 - The optimum solution: find the smallest W such that VC_W is affirmative.

Static Min-Max VC is polynomial-time solvable

- VC_W : Given an integer W , determine if G has a vertex cover of maximum weight $\leq W$.
 - Pick all vertices of weight $\leq W$ and see if this is a vertex cover.
 - The optimum solution: find the smallest W such that VC_W is affirmative.
 - Check all values W in $\{w(v) : v \in V(G)\}$.

Main Contribution (3/8)

$[A, B]$ -Gap- \mathcal{P}

- Given $0 \leq A < B \leq 1$.
- The decision problem where given an instance of \mathcal{P} such that $|\mathbf{x}_{opt}| \leq An$ or $|\mathbf{x}_{opt}| \geq Bn$.
- **Goal:** Decide whether $|\mathbf{x}_{opt}| < Bn$.

Main Theorem I

Assume that $[A, B]$ -Gap- \mathcal{P} is NP-complete, for $0 \leq A < B \leq 1$. Then for every $\alpha < \frac{B}{A}$, there is **no** (randomized) polynomial-time vanishing α -regret algorithm for online min-max- \mathcal{P} **unless** $\text{NP} = \text{RP}$.

Main Contribution (4/8)

Corollary 1

- The online min-max vertex cover problem does not admit a polynomial time vanishing $(\sqrt{2} - \epsilon)$ -regret algorithm unless $\text{NP} = \text{RP}$.
- It does not admit a polynomial time vanishing $(2 - \epsilon)$ -regret algorithm unless Unique Game is in RP .

Corollary 2

If a cardinality problem \mathcal{P} is NP-complete, then there is no polynomial time vanishing regret algorithm for online min-max- \mathcal{P} unless $\text{NP} = \text{RP}$.

- Set $\alpha = 1, A = \frac{k}{n}, B = \frac{k+1}{n} = A + \frac{1}{n}$
deciding if $|\mathbf{x}_{opt}| \leq k \Leftrightarrow$ deciding if $|\mathbf{x}_{opt}| \leq An$ or $|\mathbf{x}_{opt}| \geq Bn$.

Main Contribution (5/8)

Algorithm 2: OGD-based algorithm for Online MinMax Vertex Cover.

- 1 Select an arbitrary fractional vertex cover $x^1 \in \mathcal{Q}$.
- 2 **for** $t = 1, 2, \dots$ **do**
- 3 Round x^t to X^t : $X_i^t = 1$ if $x_i^t \geq 1/2$ and $X_i^t = 0$ otherwise.
- 4 Play $X^t \in \{0, 1\}^n$. Observe w^t (weights of vertices) and incur the cost $f^t(X^t) = \max_i w_i^t X_i^t$.
- 5 Update $y^{t+1} = x^t - \frac{1}{\sqrt{t}} g^t(x^t)$.
- 6 Project y^{t+1} to \mathcal{Q} w.r.t the ℓ_2 -norm: $x^{t+1} = \text{Proj}_{\mathcal{Q}}(y^{t+1}) := \arg \min_{x \in \mathcal{Q}} \|y^{t+1} - x\|_2$.

- We consider the relaxation:

$$\min_{x \in \mathcal{Q}} \max_{i \in V} w_i x_i,$$

- $\mathcal{Q} := \{x : x_i + x_j \leq 1, \forall (i, j) \in E, 0 \leq x_i \leq 1, \forall i \in V\}$.
- a sub-gradient $g^t(x^t) = [0, 0, \dots, w_i^t, 0, \dots, 0]$ with w_i in coordinate $\arg \max_{1 \leq j \leq n} w_j^t x_j^t$ and 0 otherwise.
- Round the solution: $X_{i+1} = 1$ if $x_i^{t+1} \geq 1/2$ and 0 otherwise.

Main Contribution (6/8)

Theorem (OGD for online Min-Max VC)

Let $W = \max_{1 \leq t \leq T} \max_{1 \leq i \leq n} w_i^t$. Then, after T steps, Algorithm 2 achieves

$$\sum_{t=1}^T \max_{1 \leq i \leq n} w_i^t X_i^t \leq 2 \cdot \min_{X^* \in \mathcal{X}} \sum_{t=1}^T \max_{1 \leq i \leq n} w_i^t X_i^* + 3W\sqrt{nT}$$

Main Contribution (7/8)

- Follow-The-Regularized-Leader (FTRL): an algorithm which is less predictable and more stable:

$$\mathbf{x}^t = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\sum_{\tau=1}^{t-1} f(\mathbf{x}, \mathbf{y}^\tau) + R(\mathbf{x}) \right),$$

where $R(\mathbf{x})$ is the regularization term.

Main Contribution (7/8)

- Follow-The-Regularized-Leader (FTRL): an algorithm which is less predictable and more stable:

$$\mathbf{x}^t = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\sum_{\tau=1}^{t-1} f(\mathbf{x}, \mathbf{y}^\tau) + R(\mathbf{x}) \right),$$

where $R(\mathbf{x})$ is the regularization term.

- Need an **optimization oracle** over the observed history.

Main Contribution (7/8)

- Follow-The-Regularized-Leader (FTRL): an algorithm which is less predictable and more stable:

$$\mathbf{x}^t = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\sum_{\tau=1}^{t-1} f(\mathbf{x}, \mathbf{y}^\tau) + R(\mathbf{x}) \right),$$

where $R(\mathbf{x})$ is the regularization term.

- Need an **optimization oracle** over the observed history.

Multi-instance version of min-max- \mathcal{P}

Given an integer $N > 0$, a set \mathcal{X} of feasible solutions, and N objective functions f_1, f_2, \dots, f_N over \mathcal{X} .

Goal: Minimize $\sum_{i=1}^N f_i(\mathbf{x})$ over \mathcal{X} .

Main Contribution (8/8)

Examples:

- Min-max vertex cover
 - Weight function $w : V \mapsto \mathbb{R}^+$ on the **vertices**.
- Min-max perfect matching
 - Weight function $w : E \mapsto \mathbb{R}^+$ on the **edges**.
 - The weight of the heaviest edge on the perfect matching is minimized.
- Min-max path
 - Given a graph $G = (V, E)$ and two vertices s, t , and a weight function $w : E \mapsto \mathbb{R}^+$ on the **edges**.
 - The weight of the heaviest edge in the s - t path is minimized.

Main Theorem II

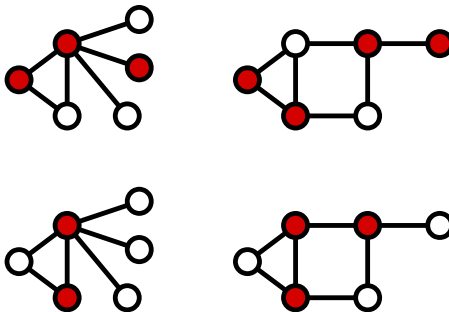
The multi-instance version of min-max perfect matching, min-max path and min-max vertex cover are strongly NP-hard.

Outline

- 1 Introduction
 - The Online Learning Framework
 - Main Contribution
- 2 Main Theorem I
 - The Proof
 - An OGD for Online Min-Max-VC
- 3 Main Theorem II
 - Multi-Instance Min-Max VC
 - Multi-Instance Min-Max Perfect Matching

Vertex Cover

Miym, CC BY-SA 3.0, via Wikimedia Commons



Proof of Main Theorem I

Main Theorem I

Assume that the problem $[A, B]$ -Gap- \mathcal{P} is NP-complete, for $0 \leq A < B \leq 1$. Then for every $\alpha < \frac{B}{A}$, there is **no** (randomized) polynomial-time vanishing α -regret algorithm for online min-max- \mathcal{P} **unless** $\text{NP} = \text{RP}$.

- Assumption: a vanishing α -regret algorithm \mathcal{O} **as an oracle** for online min-max- \mathcal{P} with $\alpha = \frac{B}{A} - \epsilon = (1 - \epsilon')\frac{B}{A}$, for $\epsilon > 0$.
- Devise a polynomial time algorithm that
 - answers 'yes' with prob. $< D < 1$ if $|\mathbf{x}_{opt}| \leq An$
 - answers 'no' if $|\mathbf{x}_{opt}| \geq Bn$.
- ★ **Note:** if $|\mathbf{x}_{opt}| \geq Bn$, all the solutions \mathbf{x}_t computed by \mathcal{O} must have size $\geq Bn$.

Algorithm for the $[A, B]$ -Gap- \mathcal{P}

- ① **for** $t = 1, 2, \dots, T$ **do**
 - Choose $\mathbf{x}^t \in \mathcal{X}$ according to the random distribution given by \mathcal{O} .
 - **if** $|\mathbf{x}^t| < Bn$ **then return** 'yes' (i.e., $|\mathbf{x}_{opt}| \leq An$).
 - Fix a weight vector w^t by assigning weight 1 to an element of \mathcal{U} chosen uniformly at random and weight 0 to all other elements.
 - Feed the weight vector and the cost $f^t(\mathbf{x}^t) = \max_{u \in \mathbf{x}^t} w^t(u)$ back to \mathcal{O} .
- ② **return** 'No' (i.e., $|\mathbf{x}_{opt}| \geq Bn$).

Proof of Main Theorem I (contd.)

- Assume that $|\mathbf{x}_{opt}| \leq An$.
- Let E be the event that the algorithm returns 'No'.
 - It finds $|\mathbf{x}_t| \geq Bn$ at each step $t \in [T]$.
- We get

$$\Pr[E] = \Pr \left[\bigcap_{t=1}^T \{|\mathbf{x}^t| \geq Bn\} \right]$$

Proof of Main Theorem I (contd.)

- Assume that $|\mathbf{x}_{opt}| \leq An$.
- Let E be the event that the algorithm returns 'No'.
 - It finds $|\mathbf{x}_t| \geq Bn$ at each step $t \in [T]$.
- We get

$$\Pr[E] = \Pr \left[\bigcap_{t=1}^T \{|\mathbf{x}^t| \geq Bn\} \right] \leq \Pr[X \geq TBn]$$

Proof of Main Theorem I (contd.)

- Assume that $|\mathbf{x}_{opt}| \leq An$.
- Let E be the event that the algorithm returns 'No'.
 - It finds $|\mathbf{x}_t| \geq Bn$ at each step $t \in [T]$.
- We get

$$\begin{aligned} \Pr[E] &= \Pr \left[\bigcap_{t=1}^T \{|\mathbf{x}^t| \geq Bn\} \right] \leq \Pr[X \geq TBn] \leq \frac{\mathbf{E}[X]}{TBn} \\ &= \frac{\sum_{t=1}^T \mathbf{E}[|\mathbf{x}^t|]}{TBn} = \frac{\sum_{t=1}^T \mathbf{E}[f^t(\mathbf{x}^t)]}{TB}. \end{aligned}$$

where $X = \sum_{t=1}^T |\mathbf{x}^t|$, and $\mathbf{E}[f^t(\mathbf{x}^t)] = \mathbf{E}[|\mathbf{x}^t|]/n$.

Proof of Main Theorem I (contd.)

Note:

- $|\mathbf{x}_{opt}| \leq An$ (by assumption).
- Only one element of weight 1 is picked uniformly at random at each time t

Hence, $\Pr[f^t(\mathbf{x}_{opt}) = 1] \leq A$

Proof of Main Theorem I (contd.)

Note:

- $|\mathbf{x}_{opt}| \leq An$ (by assumption).
- Only one element of weight 1 is picked uniformly at random at each time t

Hence, $\Pr[f^t(\mathbf{x}_{opt}) = 1] \leq A \Rightarrow \sum_{t=1}^T \mathbf{E}[f^t(\mathbf{x}_{opt})] \leq AT$.

- Since \mathcal{O} is a vanishing α -regret algorithm with $\alpha = (1 - \epsilon') \frac{B}{A}$,

Proof of Main Theorem I (contd.)

Note:

- $|\mathbf{x}_{opt}| \leq An$ (by assumption).
- Only one element of weight 1 is picked uniformly at random at each time t

Hence, $\Pr[f^t(\mathbf{x}_{opt}) = 1] \leq A \Rightarrow \sum_{t=1}^T \mathbf{E}[f^t(\mathbf{x}_{opt})] \leq AT$.

- Since \mathcal{O} is a vanishing α -regret algorithm with $\alpha = (1 - \epsilon')\frac{B}{A}$,

$$\begin{aligned} \sum_{t=1}^T \mathbf{E}[f^t(\mathbf{x}^t)] &\leq \alpha \sum_{t=1}^T \mathbf{E}[f^t(\mathbf{x}_{opt})] + \text{poly}(n) T^c \\ &\leq (1 - \epsilon')BT + \text{poly}(n) T^c. \end{aligned}$$

Proof of Main Theorem I (contd.)

Hence,

$$\Pr[E] \leq \frac{(1 - \epsilon')BT + \text{poly}(n)T^c}{BT} = (1 - \epsilon') + \frac{\text{poly}(n)T^{c-1}}{B}.$$

Proof of Main Theorem I (contd.)

Hence,

$$\Pr[E] \leq \frac{(1 - \epsilon')BT + \text{poly}(n)T^c}{BT} = (1 - \epsilon') + \frac{\text{poly}(n)T^{c-1}}{B}.$$

We can choose $T = \left(\frac{B\epsilon'}{2\text{poly}(n)}\right)^{\frac{1}{c-1}} = \left(\frac{A\epsilon}{2\text{poly}(n)B}\right)^{\frac{1}{c-1}}$, then

$$\Pr[E] \leq 1 - \frac{\epsilon'}{2} = 1 - \frac{A\epsilon}{2B}.$$

(constant; strictly smaller than 1)

Proof of Main Theorem I (contd.)

Hence,

$$\Pr[E] \leq \frac{(1 - \epsilon')BT + \text{poly}(n)T^c}{BT} = (1 - \epsilon') + \frac{\text{poly}(n)T^{c-1}}{B}.$$

We can choose $T = \left(\frac{B\epsilon'}{2\text{poly}(n)}\right)^{\frac{1}{c-1}} = \left(\frac{A\epsilon}{2\text{poly}(n)B}\right)^{\frac{1}{c-1}}$, then

$$\Pr[E] \leq 1 - \frac{\epsilon'}{2} = 1 - \frac{A\epsilon}{2B}.$$

(constant; strictly smaller than 1)

- We've (roughly) shown that the $[A, B]$ -Gap- \mathcal{P} is in RP.

The hardness result for online Min-Max VC is tight

Algorithm 2: OGD-based algorithm for Online MinMax Vertex Cover.

- 1 Select an arbitrary fractional vertex cover $x^1 \in \mathcal{Q}$.
- 2 **for** $t = 1, 2, \dots$ **do**
- 3 Round x^t to X^t : $X_i^t = 1$ if $x_i^t \geq 1/2$ and $X_i^t = 0$ otherwise.
- 4 Play $X^t \in \{0, 1\}^n$. Observe w^t (weights of vertices) and incur the cost $f^t(X^t) = \max_i w_i^t X_i^t$.
- 5 Update $y^{t+1} = x^t - \frac{1}{\sqrt{t}} g^t(x^t)$.
- 6 Project y^{t+1} to \mathcal{Q} w.r.t the ℓ_2 -norm: $x^{t+1} = \text{Proj}_{\mathcal{Q}}(y^{t+1}) := \arg \min_{x \in \mathcal{Q}} \|y^{t+1} - x\|_2$.

Theorem (OGD for online Min-Max VC)

Let $W = \max_{1 \leq t \leq T} \max_{1 \leq i \leq n} w_i^t$. Then, after T steps, Algorithm 2 achieves

$$\sum_{t=1}^T \max_{1 \leq i \leq n} w_i^t X_i^t \leq 2 \cdot \min_{X^* \in \mathcal{X}} \sum_{t=1}^T \max_{1 \leq i \leq n} w_i^t X_i^* + 3W\sqrt{nT}$$

Proof of the tightness

- The guarantee from the OGD algorithm:

$$\sum_{t=1}^T \max_{1 \leq i \leq n} w_i^t \mathbf{x}_i^t \leq \min_{X^* \in \mathcal{Q}} \sum_{t=1}^T \max_{1 \leq i \leq n} w_i^t \mathbf{x}_i^* + \frac{3DG}{2} \sqrt{T}$$

- $D \leq \sqrt{n}$ (diameter of \mathcal{Q}).
- $G \leq W$: Lipschitz constant of g^t .
- $\max_{1 \leq i \leq n} \mathbf{x}_i^t w_i^t \leq 2 \max_{1 \leq i \leq n} \mathbf{x}_i^t w_i^t$ by the rounding procedure.

Outline

- 1 Introduction
 - The Online Learning Framework
 - Main Contribution
- 2 Main Theorem I
 - The Proof
 - An OGD for Online Min-Max-VC
- 3 Main Theorem II
 - Multi-Instance Min-Max VC
 - Multi-Instance Min-Max Perfect Matching

Recall Main Theorem II

- Follow-The-Regularized-Leader (FTRL): an algorithm which is less predictable and more stable:

$$\mathbf{x}^t = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\sum_{\tau=1}^{t-1} f(\mathbf{x}, \mathbf{y}^\tau) + R(\mathbf{x}) \right),$$

where $R(\mathbf{x})$ is the regularization term.

Recall Main Theorem II

- Follow-The-Regularized-Leader (FTRL): an algorithm which is less predictable and more stable:

$$\mathbf{x}^t = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\sum_{\tau=1}^{t-1} f(\mathbf{x}, \mathbf{y}^\tau) + R(\mathbf{x}) \right),$$

where $R(\mathbf{x})$ is the regularization term.

- Need an **optimization oracle** over the observed history.

Recall Main Theorem II

- Follow-The-Regularized-Leader (FTRL): an algorithm which is less predictable and more stable:

$$\mathbf{x}^t = \arg \min_{\mathbf{x} \in \mathcal{X}} \left(\sum_{\tau=1}^{t-1} f(\mathbf{x}, \mathbf{y}^\tau) + R(\mathbf{x}) \right),$$

where $R(\mathbf{x})$ is the regularization term.

- Need an **optimization oracle** over the observed history.

Multi-instance version of min-max- \mathcal{P}

Given an integer $N > 0$, a set \mathcal{X} of feasible solutions, and N **objective functions** f_1, f_2, \dots, f_N over \mathcal{X} .

Goal: Minimize $\sum_{i=1}^N f_i(\mathbf{x})$ over \mathcal{X} .

Remark

- The problems \mathcal{P} could be polynomially solvable when using a “sum” objective.
 - Main Theorem I cannot be applied.

Remark

- The problems \mathcal{P} could be polynomially solvable when using a “sum” objective.
 - Main Theorem I cannot be applied.
- Main Theorem II shows that FTRL fail to efficiently solve the online min-max- \mathcal{P} .

Multi-Instance Min-Max VC

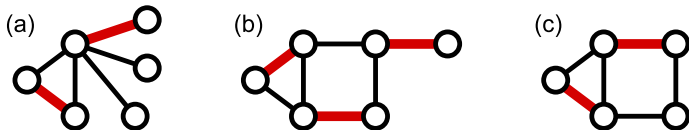
- A straightforward reduction from VC (since VC is strongly NP-hard).
- Let say $V = \{v_1, v_2, \dots, v_n\}$.
Construct n weight functions $w^1, w^2, \dots, w_n : V \mapsto \mathbb{R}$ such that
 - In w^i : we set $w^i(v_i) = 1$ and $w(v) = 0$ for $v \neq v_i$.

Multi-Instance Min-Max VC

- A straightforward reduction from VC (since VC is strongly NP-hard).
- Let say $V = \{v_1, v_2, \dots, v_n\}$.
Construct n weight functions $w^1, w^2, \dots, w_n : V \mapsto \mathbb{R}$ such that
 - In w^i : we set $w^i(v_i) = 1$ and $w(v) = 0$ for $v \neq v_i$.
- Any vertex cover has total cost equal to its size.

Perfect Matching

Miym, CC BY-SA 3.0, via Wikimedia Commons



- Maximum cardinality matchings.
- Only in (b) there is a perfect matching.

Multi-Instance Min-Max Perfect Matching (1/3)

- Reduction from the Max-3-DNF problem.
 - A 3-DNF formula: $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_3 \wedge x_4)$.
 - $(x_1 \wedge x_2 \wedge x_3)$: a clause
 - x_1 or $\neg x_2$: literals

Multi-Instance Min-Max Perfect Matching (1/3)

- Reduction from the Max-3-DNF problem.
 - A 3-DNF formula: $(x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge \neg x_2 \wedge \neg x_3) \vee (x_1 \wedge x_3 \wedge x_4)$.
 - $(x_1 \wedge x_2 \wedge x_3)$: a clause
 - x_1 or $\neg x_2$: literals
- Given
 - n Boolean variables $X = \{x_1, x_2, \dots, x_n\}$
 - m clauses C_1, C_2, \dots, C_m (conjunctions of 3 literals of X)

Goal: Determine a truth assignment $\sigma : X \mapsto \{T, F\}$ such that the number of satisfied clauses is maximized.

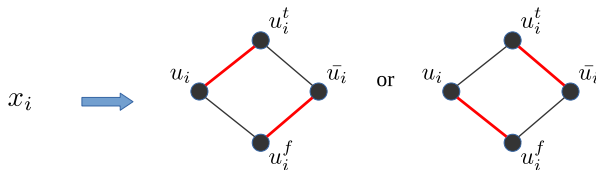
Multi-Instance Min-Max Perfect Matching (2/3)

For every instance \mathcal{I} of Max-3-DNF, construct a graph $G(V, E)$ and m weight functions as follows.

Multi-Instance Min-Max Perfect Matching (2/3)

For every instance \mathcal{I} of Max-3-DNF, construct a graph $G(V, E)$ and m weight functions as follows.

- Each x_i is associated a 4-cycle on vertices $(u_i, u_i^t, \bar{u}_i, u_i^f)$.



- Weight function corresponds to clause C_j :
 - $w^j(u_i u_i^t) = 1$ if $\neg x_i \in C_j$, otherwise $w^j(u_i u_i^t) = 0$.
Edges incident to vertices \bar{u}_i always get weight 0.
- ★ The instance \mathcal{I}' of multi-instance min-max matching is constructed (in polynomial time).

Multi-Instance Min-Max Perfect Matching (3/3)

- A truth assignment σ of \mathcal{I} corresponds to a matching M_σ of G .
- $\text{value}(\mathcal{I}, \sigma) = m - \text{value}(\mathcal{I}', M_\sigma)$

Multi-Instance Min-Max Perfect Matching (3/3)

- A truth assignment σ of \mathcal{I} corresponds to a matching M_σ of G .
- $\text{value}(\mathcal{I}, \sigma) = m - \text{value}(\mathcal{I}', M_\sigma)$
- Assume that there exists a $(1 + \epsilon)$ -approximation algorithm for multi-instance min-max perfect matching, then we can get a $(1 - \rho\epsilon)$ approximation algorithm for Max-3-DNF for some constant ρ .

Multi-Instance Min-Max Perfect Matching (3/3)

- A truth assignment σ of \mathcal{I} corresponds to a matching M_σ of G .
- $\text{value}(\mathcal{I}, \sigma) = m - \text{value}(\mathcal{I}', M_\sigma)$
- Assume that there exists a $(1 + \epsilon)$ -approximation algorithm for multi-instance min-max perfect matching, then we can get a $(1 - \rho\epsilon)$ approximation algorithm for Max-3-DNF for some constant ρ .
- Thus, multi-instance min-max perfect matching is APX-hard.

Discussion