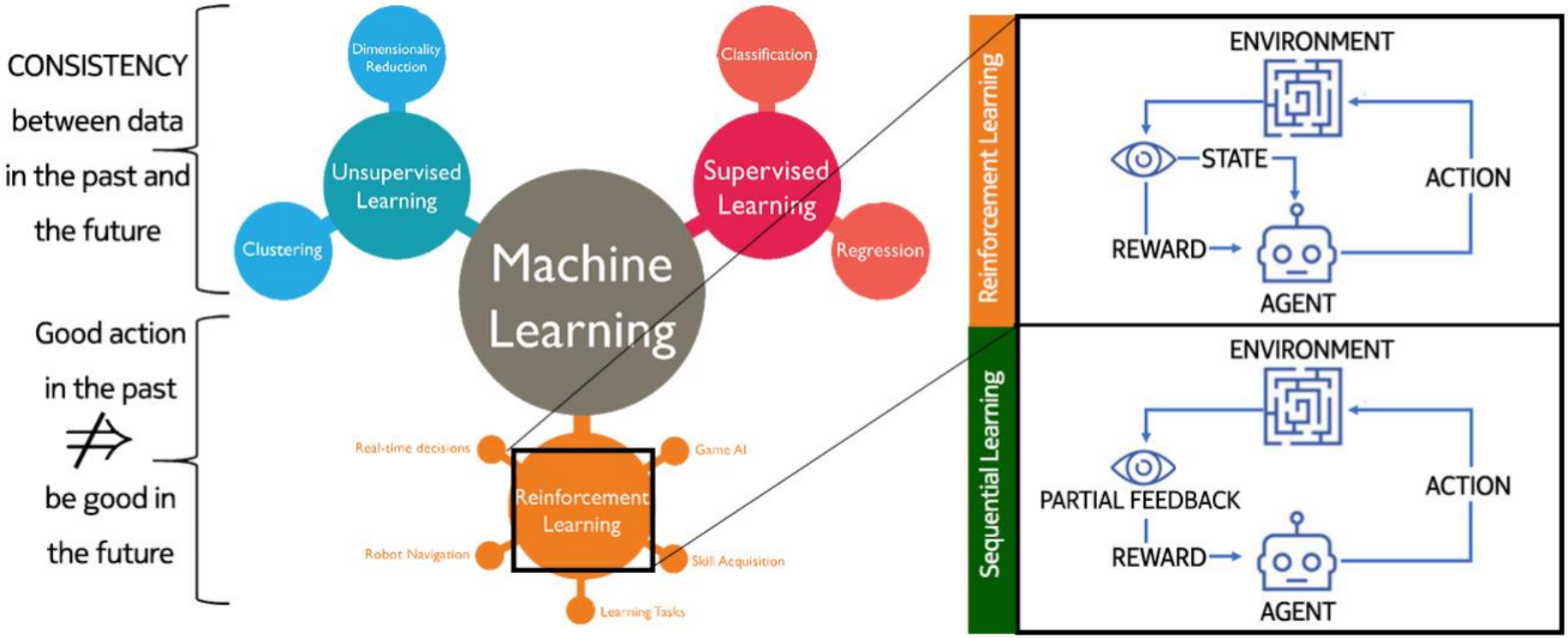
# Learning in Path Planning Problem with Side-Observations - An Instance of Multi-armed Bandit Problems

Dong Quan VU
Nokia Bell Labs France, AAAID Team & LIG, POLARIS

joint work with Patrick Loiseau (LIG & MPI-SWS), Alonso Silva (Safran Tech), Long-Thanh Tran (ESC- University of Southampton)

June 2019

# Online/Sequential Learning




**Online Learning**

**for** $t = 1, 2, \ldots$
- receive question $\mathbf{x}_t \in \mathcal{X}$
- predict $p_t \in D$
- receive true answer $y_t \in \mathcal{Y}$
- suffer loss $l(p_t, y_t)$

- Data arrives one by one
- A decision has to be made for each new piece of data
- Get some feedback
- Update the decision rule
- Feedback may depend on decision

# Stochastic Multi-armed Bandit

A simple stochastic model:

$$\forall k = 1, \dots, K, \quad (X_{k,t})_{t \in \mathbb{N}} \quad \text{is} \quad \text{i.i.d.} \quad \text{with a distribution } \nu_k$$

$$K \text{ arms} \leftrightarrow K \text{ (unknown) probability distribution}$$


$\nu_1$ $\nu_2$ $\nu_3$ $\nu_4$ $\nu_5$

At round $t$, an agent:

- chooses an arm $A_t$
- observes a sample $X_t = X_{A_t,t} \sim \nu_{A_t}$ (loss)

The sampling strategy (or bandit algorithm) $(A_t)$ is sequential:

$$A_{t+1} = F_t(A_1, X_1, \dots, A_t, X_t).$$

# Stochastic Multi-armed Bandit

A simple stochastic model:

$$\forall k = 1, \dots, K, \quad (X_{k,t})_{t \in \mathbb{N}} \quad \text{is} \quad \text{i.i.d.} \quad \text{with a distribution } \nu_k$$

$$K \text{ arms} \leftrightarrow K \text{ (unknown) probability distribution}$$


$\mu_1$ $\mu_2$ $\mu_3$ $\mu_4$ $\mu_5$

**Several possible goals**:

- find quickly the arm with smallest mean (optimal exploration)
- Minimize cumulative losses $\mathbb{E}\left[\sum_{t=1}^{T} X_t\right]$ (exploration/exploitation tradeoff)
- (more general) learn quickly *something* about the distributions $\nu_k$

# Why Bandits ?


$\nu_1$ $\nu_2$ $\nu_3$ $\nu_4$ $\nu_5$

**Goal:** maximize ones' gains in a casino ?

(HOPELESS)

## Clinical trials

Historical motivation [Thompson 1933]





$\mathcal{B}(\mu_1)$ $\mathcal{B}(\mu_2)$ $\mathcal{B}(\mu_3)$ $\mathcal{B}(\mu_4)$ $\mathcal{B}(\mu_5)$

For the $t$-th patient in a clinical study,

- chooses a treatment $A_t$
- observes a response $X_t \in \{0, 1\} : \mathbb{P}(X_t = 1) = \mu_{A_t}$

**Goal:** identify the best treatment / maximize the number of patients healed

$$ Modern motivation
(recommender systems, online advertisement, A/B Testing...)


$\nu_1$ $\nu_2$ $\nu_3$ $\nu_4$ $\nu_5$

For the $t$-th visitor of a website,

- recommend a movie $A_t$
- observe a rating $X_t \sim \nu_{A_t}$ (e.g. $X_t \in \{1, \dots, 5\}$)

**Goal:** maximize the sum of ratings

## Cognitive radio transmission

**Agent**: a smart radio device
**Arms**: radio channels (frequency bands)

*streams indicating channel availabilities*

| | | | | | | |
|---|---|---|---|---|---|---|
| Channel 1 | $X_{1,1}$ | $X_{1,2}$ | $\dots$ | $X_{1,t}$ | $\dots$ | $X_{1,T}$ |
| Channel 2 | $X_{2,1}$ | $X_{2,2}$ | $\dots$ | $X_{2,t}$ | $\dots$ | $X_{2,T}$ |
| $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | $\dots$ | |
| Channel $K$ | $X_{K,1}$ | $X_{K,2}$ | $\dots$ | $X_{K,t}$ | $\dots$ | $X_{K,T}$ |

At round $t$, the device:

- selects channel $A_t$
- observes the channel availability $X_t = X_{A_t,t} = 0$ or 1

**Goal:** Maximize the number of sucessfull transmissions

# Adversarial Bandit

- Change notation: Actions $\boldsymbol{p} \in \mathcal{P}$ ($K := |\mathcal{P}|$). Loss $L_t(\boldsymbol{p})$ instead of Reward.
- No i.i.d assumption on the reward/loss generation.
- Oblivious adversary: the loss $L_t$ does not depends on actions $\tilde{\boldsymbol{p}}_s, s \leq t$ in previous stage i.e., the sequence of losses $L_1, L_2, \cdots, L_T$ is chosen in advance.
- Minimizing the expected regret:

$$R_T := \mathbb{E}\left[\sum\nolimits_{t\in[T]} L_t\left(\tilde{\boldsymbol{p}}_t\right)\right] - \min_{\boldsymbol{p}^*\in\mathcal{P}} \sum\nolimits_{t\in[T]} L_t\left(\boldsymbol{p}^*\right)$$

$$= \text{expected cumulative loss} - \text{cumulative loss of best action in hindsight}$$

* Deterministic policy is doom in worst-case scenario $\Rightarrow$ Randomized policy = trade-off between exploration and exploitation. Target: sub-linear regret

$$u_1 = \begin{pmatrix} \frac{1}{2} \\ 0 \end{pmatrix}, \quad u_2 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad u_3 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad u_4 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad u_5 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad \cdots$$

$$R_T \sim \frac{T}{2}$$

# Adversarial Bandit

- Change notation: Actions $\boldsymbol{p} \in \mathcal{P}$ ($K := |\mathcal{P}|$). Loss $L_t(\boldsymbol{p})$ instead of Reward.
- No i.i.d assumption on the reward/loss generation.
- Oblivious adversary: the loss $L_t$ does not depends on actions $\tilde{\boldsymbol{p}}_s, s \leq t$ in previous stage i.e., the sequence of losses $L_1, L_2, \cdots, L_T$ is chosen in advance.
- Minimizing the expected regret:

$$R_T := \mathbb{E}\left[\sum\nolimits_{t\in[T]} L_t\left(\tilde{\boldsymbol{p}}_t\right)\right] - \min_{\boldsymbol{p}^*\in\mathcal{P}} \sum\nolimits_{t\in[T]} L_t\left(\boldsymbol{p}^*\right)$$

$$= \text{expected cumulative loss} - \text{cumulative loss of best action in hindsight}$$

# Adversarial Bandit

- Change notation: Actions $\boldsymbol{p} \in \mathcal{P}$ ($K := |\mathcal{P}|$). Loss $L_t(\boldsymbol{p})$ instead of Reward.
- No i.i.d assumption on the reward/loss generation.
- Oblivious adversary: the loss $L_t$ does not depends on actions $\tilde{\boldsymbol{p}}_s, s \leq t$ in previous stage i.e., the sequence of losses $L_1, L_2, \cdots, L_T$ is chosen in advance.
- Minimizing the expected regret:

$$R_T := \mathbb{E}\left[\sum\nolimits_{t\in[T]} L_t\left(\tilde{\boldsymbol{p}}_t\right)\right] - \min_{\boldsymbol{p}^*\in\mathcal{P}} \sum\nolimits_{t\in[T]} L_t\left(\boldsymbol{p}^*\right)$$

$$= \text{expected cumulative loss} - \text{cumulative loss of best action in hindsight}$$

* Deterministic policy is doom in worst-case scenario $\Rightarrow$ Randomized policy = trade-off between exploration and exploitation.

Expected Regret's Bounds

| | $K$-Arm | |
|---|---|---|
| | Full | Partial |
| Lower Bound | $\sqrt{T \ln K}$ | $\sqrt{TK}$ |
| Upper Bound | $\sqrt{T \ln K}$ | $\sqrt{TK}$ |

# Combinatorial Bandit

At each time $t$,

- A loss vector $\boldsymbol{\ell}_t \in [0,1]^n$ is adversarially chosen.
- Unknowing this, the learner chooses a vector $\tilde{\boldsymbol{p}}_t \in \mathcal{P} \subset \{0,1\}^n$ (note that $|\mathcal{P}| = \Omega(2^n)$).
- The learner suffers the loss $L_t := \langle \tilde{\boldsymbol{p}}_t, \boldsymbol{\ell}_t \rangle = \sum_{i \in [n]} \boldsymbol{\ell}_t(\tilde{\boldsymbol{p}}_t)[i] \cdot \tilde{\boldsymbol{p}}_t[i] \in \mathbb{R}$.
- Feedback on losses are observed:
  - Full information: Observe the whole vector $\boldsymbol{\ell}_t \Rightarrow$ know losses of all actions even the non-chosen ones.
  - Bandit feedback: Knows only the scalar $L_t \in \mathcal{R}$.
  - Semi-bandit feedback: Knows the products $\boldsymbol{\ell}_t(\tilde{\boldsymbol{p}}_t)[i] \cdot \tilde{\boldsymbol{p}}_t[i]$, i.e., knows the losses corresponding to the element where $\tilde{\boldsymbol{p}}_t[i] = 1$.

Motivation: Routing/Path Planning Problems, Multi-task Learning, Perfect Matching, Balanced Cuts, etc.,

Naive idea: applying MAB algorithms (say, Exp2) we obtain:

- $R_T \leq \mathcal{O}(\sqrt{T|\mathcal{P}|})$ (Very Large Bound).
- Issues with the complexity of the algorithms.

# Combinatorial Bandit

At each time $t$,

- A loss vector $\boldsymbol{\ell}_t \in [0,1]^n$ is adversarially chosen.
- Unknowing this, the learner chooses a vector $\tilde{\boldsymbol{p}}_t \in \mathcal{P} \subset \{0,1\}^n$ (note that $|\mathcal{P}| = \Omega(2^n)$).
- The learner suffers the loss $L_t := \langle \tilde{\boldsymbol{p}}_t, \boldsymbol{\ell}_t \rangle = \sum_{i \in [n]} \boldsymbol{\ell}_t(\tilde{\boldsymbol{p}}_t)[i] \cdot \tilde{\boldsymbol{p}}_t[i] \in \mathbb{R}$.
- Feedback on losses are observed:
  - Full information: Observe the whole vector $\boldsymbol{\ell}_t \Rightarrow$ know losses of all actions even the non-chosen ones.
  - Bandit feedback: Knows only the scalar $L_t \in \mathcal{R}$.
  - Semi-bandit feedback: Knows the products $\boldsymbol{\ell}_t(\tilde{\boldsymbol{p}}_t)[i] \cdot \tilde{\boldsymbol{p}}_t[i]$, i.e., knows the losses corresponding to the element where $\tilde{\boldsymbol{p}}_t[i] = 1$.

Motivation: Routing/Path Planning Problems, Multi-task Learning, Perfect Matching, Balanced Cuts, etc.,

Naive idea: applying MAB algorithms (say, Exp2) we obtain:

- $R_T \leq \mathcal{O}(\sqrt{T|\mathcal{P}|})$ (Very Large Bound).
- Issues with the complexity of the algorithms.

# Combinatorial Bandit

At each time $t$,

- A loss vector $\boldsymbol{\ell}_t \in [0,1]^n$ is adversarially chosen.
- Unknowing this, the learner chooses a vector $\tilde{\boldsymbol{p}}_t \in \mathcal{P} \subset \{0,1\}^n$ (note that $|\mathcal{P}| = \Omega(2^n)$).
- The learner suffers the loss $L_t := \langle \tilde{\boldsymbol{p}}_t, \boldsymbol{\ell}_t \rangle = \sum_{i \in [n]} \boldsymbol{\ell}_t(\tilde{\boldsymbol{p}}_t)[i] \cdot \tilde{\boldsymbol{p}}_t[i] \in \mathbb{R}$.
- Feedback on losses are observed:
  - Full information: Observe the whole vector $\boldsymbol{\ell}_t \Rightarrow$ know losses of all actions even the non-chosen ones.
  - Bandit feedback: Knows only the scalar $L_t \in \mathcal{R}$.
  - Semi-bandit feedback: Knows the products $\boldsymbol{\ell}_t(\tilde{\boldsymbol{p}}_t)[i] \cdot \tilde{\boldsymbol{p}}_t[i]$, i.e., knows the losses corresponding to the element where $\tilde{\boldsymbol{p}}_t[i] = 1$.

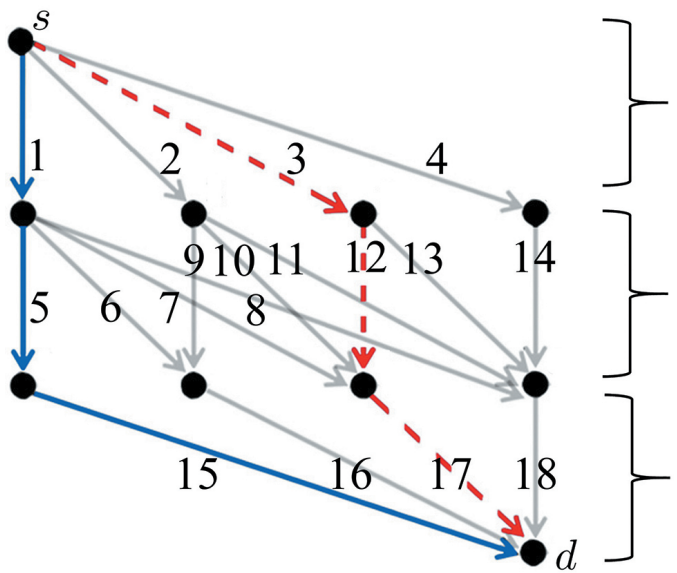
Motivation: Routing/Path Planning Problems, Multi-task Learning, Perfect Matching, Balanced Cuts, etc.,
Naive idea: applying MAB algorithms (say, Exp2) we obtain:

- $R_T \leq \mathcal{O}(\sqrt{T|\mathcal{P}|})$ (Very Large Bound).
- Issues with the complexity of the algorithms.

# The Path Planning Problem

**Path planning problem (PPP):**

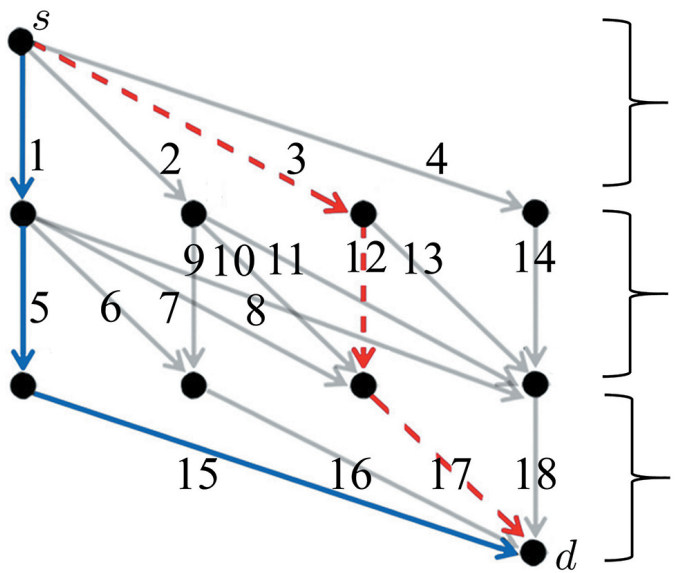
- Directed acyclic graph $G = (V, \mathcal{E})$; $\mathcal{P}$ set of paths from source to sink.
- At each stage $t \in [T]$:
  - Edge $e \in \mathcal{E}$ is embedded with a loss $\ell_t(e) \in [0, 1]$ (unknown to learner),
  - Learner chooses a path $\tilde{\boldsymbol{p}}_t \in \mathcal{P}$ and suffers sum of edges' losses: $L_t(\tilde{\boldsymbol{p}}_t) = \sum_{e \in \tilde{\boldsymbol{p}}_t} \ell_t(e)$.
  - Feedback setting: Full-info, semi-bandit, bandit.



# The Path Planning Problem

**Path planning problem (PPP):**

- Directed acyclic graph $G = (V, \mathcal{E})$; $\mathcal{P}$ set of paths from source to sink.
- At each stage $t \in [T]$:
  - Edge $e \in \mathcal{E}$ is embedded with a loss $\ell_t(e) \in [0, 1]$ (unknown to learner),
  - Learner chooses a path $\tilde{\boldsymbol{p}}_t \in \mathcal{P}$ and suffers sum of edges' losses: $L_t(\tilde{\boldsymbol{p}}_t) = \sum_{e \in \tilde{\boldsymbol{p}}_t} \ell_t(e)$.
  - Feedback setting: Full-info, semi-bandit, bandit.



# Bandit Feedback: the ComBand Benchmark

**State-of-the-art**: ComBand algorithm[1] can be applied to PPP: at stage $t$,

- Keep weight for each path and update according to *exponential multiplicative* rule;
- Sample a path based on normalized weights mixed with *exploration distribution* $\mu$;
- Estimate the loss for each path (*unbiased estimator*) based on the *co-occurrence matrix* $C_t$. Use this loss estimator to update weights.
  - $C \in \mathbb{M}_{E \times E}$ and $C_{i,j}$ is the probability that the chosen path contains edge $i$ and $j$

**Regret guarantee**: $R_T \leq \mathcal{O}(\sqrt{TE \ln(|\mathcal{P}|) + nT \ln(|\mathcal{P}|)/\lambda_{\min}})$

- $n$: length of the longest path
- $\lambda_{\min}$: smallest non-zero eigenvalue of co-occurrence matrix generated from $\mu$

**Main issues**:

1. **Running time is exponential** in $n$ (polynomial in $|\mathcal{P}|$ and $T$).
2. **Which exploration distribution** $\mu$ to optimizes $\lambda_{\min}$?
   (Standard uniform distribution can have $\lambda_{\min}$, of the order of $1/|\mathcal{P}|$).

[1] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. Journal of Computer and System Sciences, 2012

# Bandit Feedback: the ComBand Benchmark

**State-of-the-art**: ComBand algorithm[1] can be applied to PPP: at stage $t$,

- Keep weight for each path and update according to *exponential multiplicative* rule;
- Sample a path based on normalized weights mixed with *exploration distribution* $\mu$;
- Estimate the loss for each path (*unbiased estimator*) based on the *co-occurrence matrix* $C_t$. Use this loss estimator to update weights.
  - $C \in \mathbb{M}_{E\times E}$ and $C_{i,j}$ is the probability that the chosen path contains edge $i$ and $j$

**Regret guarantee**: $R_T \leq \mathcal{O}(\sqrt{TE\ln(|\mathcal{P}|) + nT\ln(|\mathcal{P}|)/\lambda_{\min}})$

- $n$: length of the longest path
- $\lambda_{\min}$: smallest non-zero eigenvalue of co-occurrence matrix generated from $\mu$

**Main issues**:

1. **Running time is exponential** in $n$ (polynomial in $|\mathcal{P}|$ and $T$).
2. **Which exploration distribution** $\mu$ to optimizes $\lambda_{\min}$?
   (Standard uniform distribution can have $\lambda_{\min}$, of the order of $1/|\mathcal{P}|$).

[1] N. Cesa-Bianchi and G. Lugosi. Combinatorial bandits. Journal of Computer and System Sciences, 2012

# Bandit Feedback: Our Algorithm EdgeCB

We propose **EdgeCB**, a novel variant of ComBand with two updates based on weight pushing (a dynamic programming technique).

Key idea: maintain weights on edges only (not paths)

- Solving Issue 1.: apply weight pushing to efficiently sample the path and compute the co-occurrence matrix (similar to CombD3[2] but with easier representation)
  $\Rightarrow$ Running time is in $\mathcal{O}(E^2T)$.
- Solving Issue 2.: propose an efficient derivative-free (black-box) optimization procedure on exploration distributions defined on the edges' weights
  $\Rightarrow$ Heuristically improves the exploration distribution.

$\Rightarrow$ Efficient algorithm
$\Rightarrow$ Same regret guarantee as ComBand but with better $\lambda_{\min}$

$$R_T \leq \mathcal{O}(\sqrt{TE\ln(|\mathcal{P}|) + nT\ln(|\mathcal{P}|)/\lambda_{\min}})$$

[2] S.Sakaue, M. Ishihata, and S.Minato. Efficient bandit combinatorial optimization algorithm with zero-suppressed binary decision diagrams. In AISTATS 2018.

# Bandit Feedback: Our Algorithm EdgeCB

We propose **EdgeCB**, a novel variant of ComBand with two updates based on weight pushing (a dynamic programming technique).

Key idea: maintain weights on edges only (not paths)

- Solving Issue 1.: apply weight pushing to efficiently sample the path and compute the co-occurrence matrix (similar to CombD3[2] but with easier representation)
  $\Rightarrow$ Running time is in $\mathcal{O}(E^2T)$.
- Solving Issue 2.: propose an efficient derivative-free (black-box) optimization procedure on exploration distributions defined on the edges' weights
  $\Rightarrow$ Heuristically improves the exploration distribution.

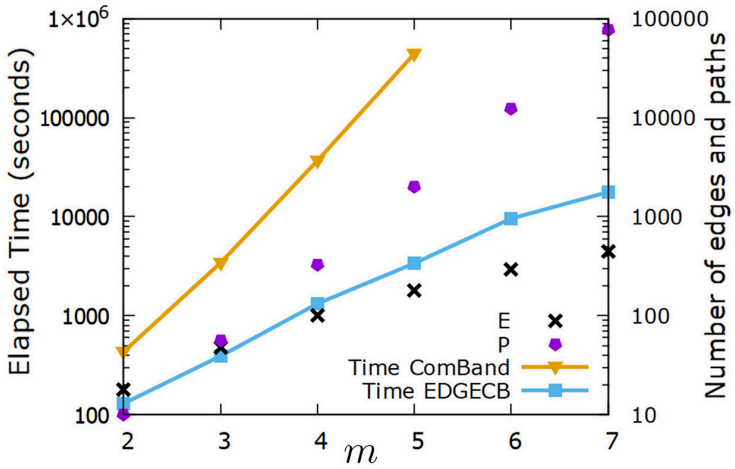
$\Rightarrow$ Efficient algorithm
$\Rightarrow$ Same regret guarantee as ComBand but with better $\lambda_{\min}$

$$R_T \leq \mathcal{O}(\sqrt{TE\ln(|\mathcal{P}|) + nT\ln(|\mathcal{P}|)/\lambda_{\min}})$$

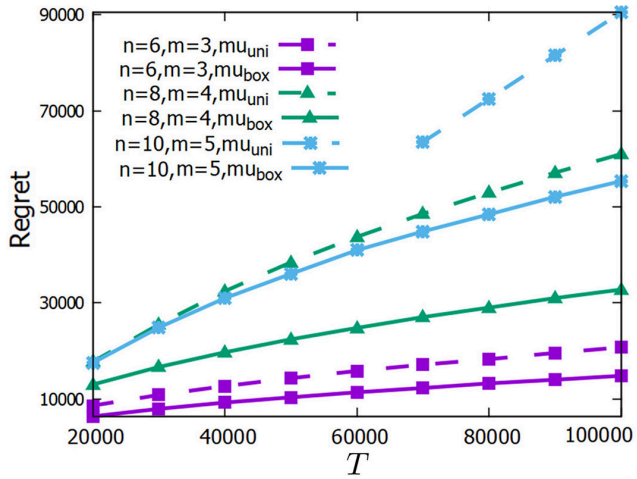
[2] S.Sakaue, M. Ishihata, and S.Minato. Efficient bandit combinatorial optimization algorithm with zero-suppressed binary decision diagrams. In AISTATS 2018.

# Numerical Experiments

- Efficiency improvement: Running time of ComBand and EdgeCB on instances where $m$ increases and $n = 2m$; $y$-axis is drawn with log-scale:




- Performance guarantee: Actual regrets of EdgeCB $(\mu_{\text{uni}})$ and EdgeCB$(\mu_{\text{box}})$.



# Semi-bandit feedback: Side Observations

**Recall**: Semi-bandit feedback: at the end of each stage, the learner observes losses of edges belonging to chosen path.

**Side observations**: from semi-bandit feedback, the learner can deduce additional feedback (at no cost)

- By observing an edge she can deduce losses of some other edges
- Observation graph $G_O^t$ (depending on $t$ and unknown at decision time):
  - Each vertex $v_e$ of $G_O^t$ corresponds to an edge $e$ of the graph $G$.
  - There is an edge from vertex $v_e$ to vertex $v_{e'}$ if in $G$, by observing the loss on edge $e$, the learner can deduce the loss on $e'$.

$\Rightarrow$ Path planning problem with semi-bandit feedback plus side-observations = SOPPP.

# Semi-bandit feedback: Side Observations

**Recall**: Semi-bandit feedback: at the end of each stage, the learner observes losses of edges belonging to chosen path.

**Side observations**: from semi-bandit feedback, the learner can deduce additional feedback (at no cost)

- By observing an edge she can deduce losses of some other edges
- Observation graph $G_O^t$ (depending on $t$ and unknown at decision time):
  - Each vertex $v_e$ of $G_O^t$ corresponds to an edge $e$ of the graph $G$.
  - There is an edge from vertex $v_e$ to vertex $v_{e'}$ if in $G$, by observing the loss on edge $e$, the learner can deduce the loss on $e'$.

$\Rightarrow$ Path planning problem with semi-bandit feedback plus side-observations = SOPPP.

# Semi bandit feedback: The FPL-IX benchmark

**State-of-the-art**: FPL-IX[3] (Follow-The-Perturbed-Leader with Implicit Exploration)

- FPL: find action that minimizing cumulative past loss estimate, perturbed
- IX: construct optimistic estimation of loss using geometric resampling

**Regret guarantee**: $R_T \leq \tilde{\mathcal{O}}(n^{3/2}\sqrt{T\alpha})$

- $\alpha$: upper bound of the independent number[4] of the observation graphs $G_O^t, t \in [T]$ (note that $\alpha \leq E$).

**Main issues**:

1. the running time of FPL-IX is only guaranteed either with high-probability or in expectation (it *can be inefficient in worst-case scenarios*).
2. FPL-IX *requires to have an efficient oracle* for solving optimization problems at each stage.

[3] Tomáš Kocák, Gergely Neu, Michal Valko, and Rémi Munos. Efficient learning by implicit exploration in bandit problems with side observations. In NIPS 2014

[4] The size of the largest independent sets that are the sets whose vertices are not connected.

# Semi-bandit Feedback: Our Algorithm Exp3-OE

We propose **Exp3-OE**, an Exp3-type algorithm solving these issues while improving the bound of FPL-IX in the CB game with semi-bandit feedback.

- Novel loss estimator (biased) based on implicit exploration with *fixed parameter* $\beta$: $\hat{\boldsymbol{\ell}}_t(e) = \boldsymbol{\ell}_t(e)/(q_t(e) + \beta)$ where $q_t(e)$ is the probability that edge $e$ is revealed at $t$.
- Weight pushing to efficiently sample a path and compute this novel loss estimator.

## Results for arbitrary SOPPP

- Exp3-OE *always* runs efficiently in $\mathcal{O}(E^3T)$ *without auxiliary oracle*.
- Exp3-OE guarantees $R_T \leq \tilde{\mathcal{O}}(n\sqrt{T\alpha \ln(|\mathcal{P}|)})$ ($\sim$ FPL-IX for general case).

# Semi-bandit Feedback: Our Algorithm Exp3-OE

We propose **Exp3-OE**, an Exp3-type algorithm solving these issues while improving the bound of FPL-IX in the CB game with semi-bandit feedback.

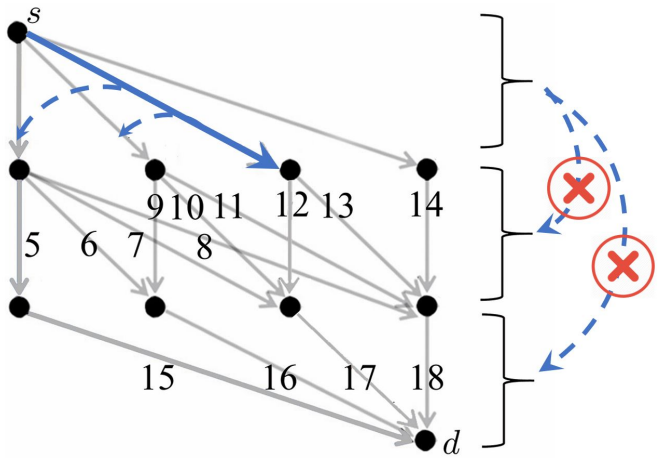
- Novel loss estimator (biased) based on implicit exploration with *fixed parameter* $\beta$: $\hat{\ell}_t(e) = \ell_t(e)/(q_t(e) + \beta)$ where $q_t(e)$ is the probability that edge $e$ is revealed at $t$.
- Weight pushing to efficiently sample a path and compute this novel loss estimator.

## Results for arbitrary SOPPP

- Exp3-OE *always* runs efficiently in $\mathcal{O}(E^3T)$ *without auxiliary oracle*.
- Exp3-OE guarantees $R_T \leq \tilde{\mathcal{O}}(n\sqrt{T\alpha \ln(|\mathcal{P}|)})$ ($\sim$ FPL-IX for general case).

# Improved Exp3-OE bounds for the Colonel Blotto Game

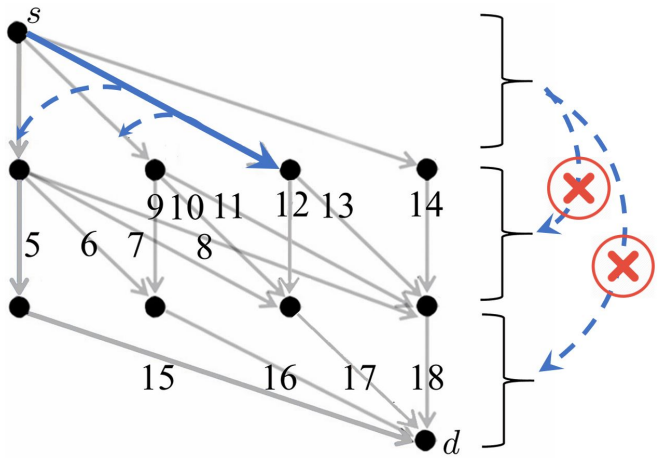
**Assumption** $(A0)$: if two edges belong to a path in $G$, then they cannot simultaneously reveal the loss of another edge.




Improve the regret upper-bound by a factor of $\sqrt{n}$.

# Improved Exp3-OE bounds for the Colonel Blotto Game

**Assumption** $(A0)$: if two edges belong to a path in $G$, then they cannot simultaneously reveal the loss of another edge.




Improve the regret upper-bound by a factor of $\sqrt{n}$.

# Summary

**Bandit feedback**:

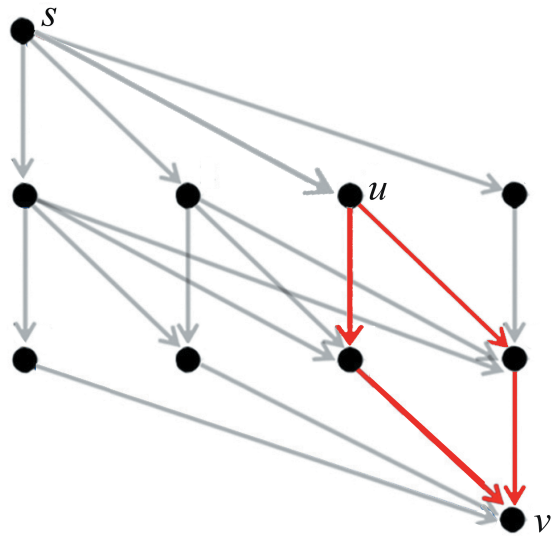
| Algorithm | Running time | Regret guarantee | Notes |
|---|---|---|---|
| ComBand | $\mathcal{O}(\exp(n)T)$ | $R_T \leq \mathcal{O}(\sqrt{T\ln(\lvert\mathcal{P}\rvert)(E+n/\lambda_{\mu_{\text{uni}}}})$ | |
| EdgeCB | $\mathcal{O}(n^2T)$ | $R_T \leq \mathcal{O}(\sqrt{T\ln(\lvert\mathcal{P}\rvert)(E+n/\lambda_{\mu_{\text{box}}}})$ | $\lambda_{\mu_{\text{box}}} \gg \lambda_{\mu_{\text{uni}}}$ |

**Semi-bandit feedback**:

| Algorithm | Running time | Regret guarantee | Notes |
|---|---|---|---|
| FPL-IX | $\text{Poly}(n)T^{3/2}$ with high-prob | $R_T \leq \tilde{\mathcal{O}}\sqrt{n^3\alpha T}$ | Require oracles |
| OSMD | $\mathcal{O}(\exp(n)T)$ | $R_T \leq \mathcal{O}\sqrt{nET}$ | Require oracles<br>Ignores side-obs |
| Exp3-OE | $\mathcal{O}(n^3T)$ | $R_T \leq \tilde{\mathcal{O}}(\sqrt{n^3\alpha T})$ | Improved if satisfy (A0) |

# Weight Pushing Technique

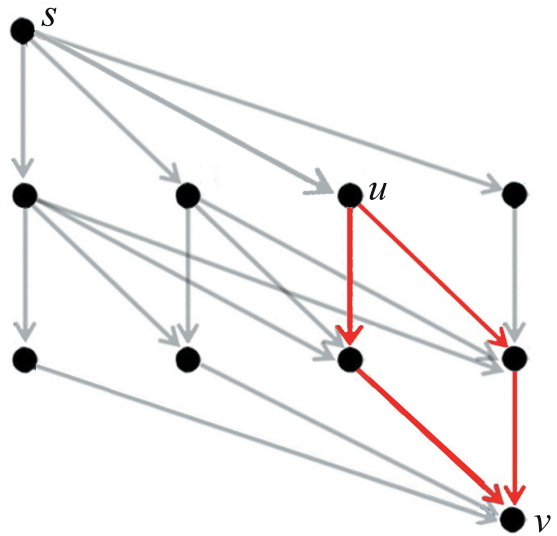
- Purpose: keeping track and updating path weights via edges ($E \ll |\mathcal{P}|$):
  - Create edges weights $w_t(e), e \in \mathcal{E}$ such that path weight = product of corresponding edges' weights.
  - Denote $H(u, v)$ the sum of paths weights from vertex $u$ to $v$.
  - Compute recursively $H(v_1, v_2), \forall v_1, v_2$ (e.g., compute $H(u, d)$ based on $H(v, d)$ for any successor vertex $v$ of $u$).
  - Computation of $H(u, v)$ for any vertex $u, v$ is done in $O(E^2)$.




- Sample a path:
  - Starting from the source $s$, sample vertex $v$ with probability $H(v, d)/H(p(v), d)$ where $p(u)$ is the predecessor of $v$; stop when reaching $d \Rightarrow$ A path is sampled out with the necessary probability distribution.
- Compute the co-occurrence matrix:
  - Entry $C_{e_1,e_2}$ of the co-occurrence matrix $C$ is the probability that $e_1, e_2$ belong to the chosen path $=$ sum of weights of paths passing through $e_1, e_2$.
- Compute the loss estimator in Exp3-OE:
  - Compute $q_t(e) :=$ probability that an edge $e$ is revealed $=$ sum of weights of paths passing through edges revealing $e$.

# Weight Pushing Technique

- Purpose: keeping track and updating path weights via edges ($E \ll |\mathcal{P}|$):
  - Create edges weights $w_t(e), e \in \mathcal{E}$ such that path weight = product of corresponding edges' weights.
  - Denote $H(u, v)$ the sum of paths weights from vertex $u$ to $v$.
  - Compute recursively $H(v_1, v_2), \forall v_1, v_2$ (e.g., compute $H(u, d)$ based on $H(v, d)$ for any successor vertex $v$ of $u$).
  - Computation of $H(u, v)$ for any vertex $u, v$ is done in $O(E^2)$.




- Sample a path:
  - Starting from the source $s$, sample vertex $v$ with probability $H(v, d)/H(p(v), d)$ where $p(u)$ is the predecessor of $v$; stop when reaching $d \Rightarrow$ A path is sampled out with the necessary probability distribution.
- Compute the co-occurrence matrix:
  - Entry $C_{e_1,e_2}$ of the co-occurrence matrix $C$ is the probability that $e_1, e_2$ belong to the chosen path $=$ sum of weights of paths passing through $e_1, e_2$.
- Compute the loss estimator in Exp3-OE:
  - Compute $q_t(e) :=$ probability that an edge $e$ is revealed $=$ sum of weights of paths passing through edges revealing $e$.

Cesa-Bianchi, N. and Lugosi, G. (2012).

Combinatorial bandits.

*Journal of Computer and System Sciences*, 78(5):1404–1422.

Sakaue, S., Ishihata, M., and Minato, S.-i. (2018).

Efficient bandit combinatorial optimization algorithm with zero-suppressed binary decision diagrams.

In *International Conference on Artificial Intelligence and Statistics*, pages 585–594.

Vu, D. Q., Loiseau, P., and Silva, A. (2019a).

Combinatorial bandits for sequential learning in colonel blotto games.

Under Review.

Vu, D. Q., Loiseau, P., Silva, A., and Tran, L.-T. (2019b).

Colonel blotto and hide-and-seek games as path planning problems with side observations.

Under review.