第21回情報論的学習理論ワークショップ (IBIS2018)

企画セッション「離散構造処理」

離散構造処理技術の機械学習への応用

石畠　正和
NTT コミュニケーション科学基礎研究所

石畠　正和（いしはた　まさかず）

- 学歴
  - 2001.4 – 2006.3 石川高専　電子情報工学科
  - 2006.4 – 2008.3 東工大　情報工学科
  - 2008.3 – 2010.3 東工大　計算工学専攻　修士課程
  - 2010.4 –2 013.4 東工大　計算工学専攻　博士課程
- 職歴
  - 2010.4 – 2013.3 日本学術振興会（DC1)
  - 2013.4 – 2016.3 NTTコミュニケーション科学基礎研究所　研究員
  - 2016.4 – 2018.3 北海道大学　大学院情報科学研究科　特任助教（出向）
  - 2018.4 – 現在　NTTコミュニケーション科学基礎研究所　研究員→研究主任
- 研究分野
  - 人工知能・機械学習
  - 特に離散構造処理技術を利用した確率モデリング・最適化など

概要

離散構造処理技術を機械学習(っぽい)問題に適用した事例を紹介

1. **“A Dynamic Programming Algorithm for Tree Trimming-based Text Summarization”**

   Nishino, Yasuda, Hirao, Minato, Nagata, NAACL’2015.

2. **“Exact Computation of Influence Spread by Binary Decision Diagrams”**

   Maehara, Suzuki, Ishihata, WWW’2017.

3. **“Efficient Bandit Combinatorial Optimization Algorithm with Zero-suppressed Binary Decision Diagrams”**

   Sakaue, Ishihata, Minato, AISTATS’2018.

4. **“Optimizing Network Reliability via Best-First Search over Decision Diagrams”**（時間があれば…）

   Nishino, Inoue, Yasuda, Minato, Nagata, INFOCOM’2018.

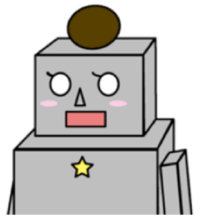
注意

- 発表時間の関係で過度に簡単に説明します。
- 実験結果なども載せますが、お気持ちだけ説明します。
- より詳細に興味がある方は論文をお読みください。
- もしくは会場にいるであろう石畠にお声がけください。

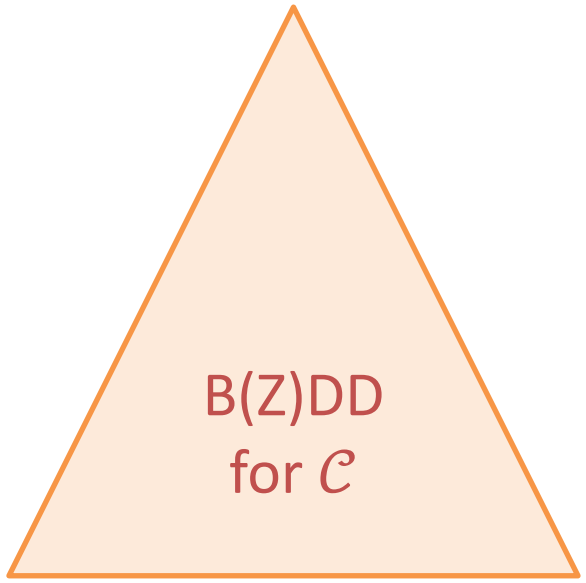
復習：BDD・ZDDがあればできること

- 台集合 V
- 制約 $\mathcal{C} \subseteq 2^V$

- 線形関数 $f(X) \coloneqq \sum_{i \in X} w_i + \sum_{i \notin X} v_i$
- 対数線形モデル $p(X) \propto \prod_{i \in X} \theta_i \prod_{i \notin X} \phi_i$



$\Delta_{\mathcal{C}}$



- 最適化 $X^* \coloneqq \mathrm{argmax}_{X \in \mathcal{C}} f(X)$
- 総和 $f(\mathcal{C}) \coloneqq \sum_{X \in \mathcal{C}} f(X)$
- 確率計算 $p(\mathcal{C}) \coloneqq \sum_{X \in \mathcal{C}} p(X)$
- サンプリング $X \sim p(X \mid \mathcal{C})$

} $O(|\Delta_{\mathcal{C}}|)$ で計算可能

[1/4]
A Dynamic Programming Algorithm for Tree Trimming-based Text Summarization

Nishino, Yasuda, Hirao, Minato, Nagata

NAACL'2015

１枚概要

- 重要文抽出
  - 与えられた文書から重要な文を取り出して要約を作る
  - 素朴に定式化するとナップザック問題
- 木刈り込みによる要約
  - 各文の依存関係を表す木が与えられ、木制約のもとで要約を作成
  - 厳密最大化手法が知られていない
- ZDDを用いた効率的な木刈り込み法を提案
  - ZDDで木制約＋ナップサック制約を表現
  - 既存手法より数百倍高速

重要文抽出

A: 桃太郎さん　桃太郎さん

お腰につけた　きび団子

一つ私に　くださいな

B: やりましょう　やりましょう

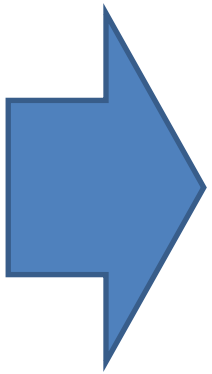
これから　鬼の征伐に

ついていくなら　やりましょう

A: 行きましょう　行きましょう

あなたについて　どこまでも

家来になって　行きましょう



A: **桃太郎さん**　桃太郎さん

お腰につけた　**きび団子**

一つ私に　**くださいな**

B: やりましょう　やりましょう

これから　**鬼の征伐に**

ついていくなら　やりましょう

A: 行きましょう　行きましょう

あなたについて　どこまでも

家来になって　行きましょう

ナップサック問題としての要約

- Input:
  - S_i ≔ 文 ($i \in [N]$)
  - ℓ_i ≔ S_i の長さ
  - w_i ≔ S_i の重要度
  - L ≔ 要約文書の長さの最大

i	S_i	ℓ_i	w_i
1	桃太郎さん	5	4
2	桃太郎さん	5	4
3	お腰につけた	6	2
4	きび団子	4	4
5	一つ私に	4	3
6	くださいな	5	5

- Output:
  - $X^* \coloneqq \mathrm{argmax}_{X \in \mathcal{C}} \sum_{i \in X} w_i$
  - $\mathcal{C} \coloneqq \{X \subseteq [N] \mid \sum_{i \in X} \ell_i \leq L\}$

☺ $O(NL)$ 時間で最大化可能

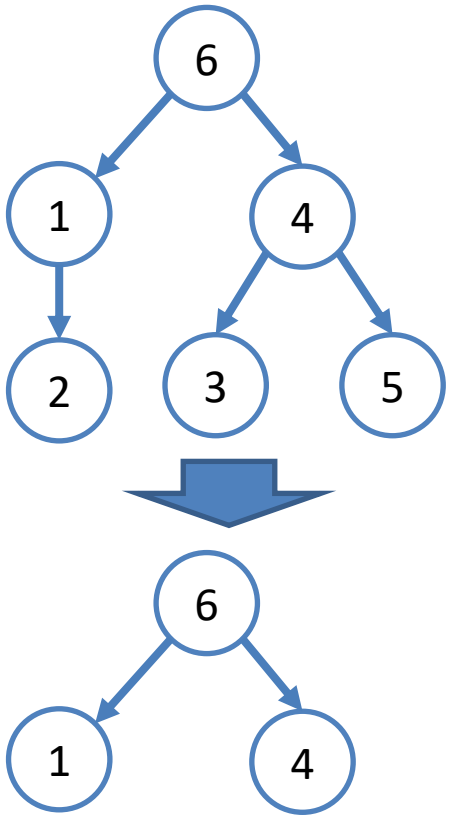
☹ 意味の重複が起こりやすい

木刈り込みとしての要約 [Hirao+ 13]

- Input:
  - ナップザック問題：$\{(S_i, \ell_i, w_i)\}_{i\in[N]}, L$
  - $T \coloneqq \langle [N], E\rangle$: S_i の依存関係を表した木

    ※ 今回はこの木がどのように得られるかは割愛

- Output:
  - $X^* \coloneqq \mathrm{argmax}_{X\in\mathcal{C}} \sum_{i\in X} w_i$
  - $\mathcal{C} \coloneqq \mathcal{C}_{\mathrm{knap}} \cap \mathcal{C}_{\mathrm{tree}}$
  - $\mathcal{C}_{\mathrm{knap}} \coloneqq \{X \subseteq [N] \mid \sum_{i\in X} \ell_i \leq L\}$
  - $\mathcal{C}_{\mathrm{tree}} \coloneqq \{X \subseteq [N] \mid T[X] \text{ is tree.}\}$



i	S_i	ℓ_i	w_i
1	桃太郎さん	5	4
2	桃太郎さん	5	4
3	お腰につけた	6	2
4	きび団子	4	4
5	一つ私に	4	3
6	くださいな	5	5

☺ 文同士の関係を考慮

☹ 効率的な厳密最大化法がない

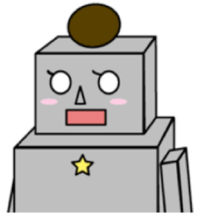
本研究

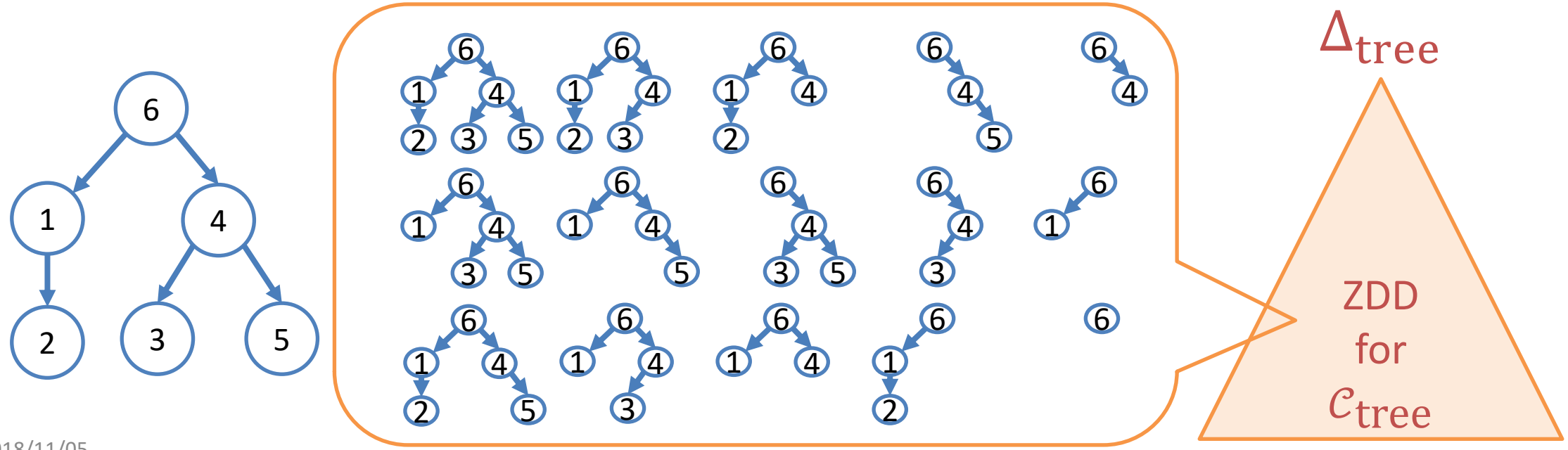
- 木刈り込み問題に対する厳密最大化法を提案
  - ZDD で制約 $\mathcal{C} := \mathcal{C}_{\mathrm{knap}} \cap \mathcal{C}_{\mathrm{tree}}$ を表現
  - ZDD 上で制約付き線形関数最大化 $X^* := \mathrm{argmax}_{X \in \mathcal{C}} \sum_{i \in X} w_i$
- 実験的に既存法より数百倍高速であることを確認
  - 既存法：整数計画問題として定式化＋整数計画ソルバ
  - ※ 整数計画問題では厳密解が得られるとは限らない

今回のDDポイント：

$\mathcal{C} \subseteq 2^{[N]}$ なので一般的には指数サイズ

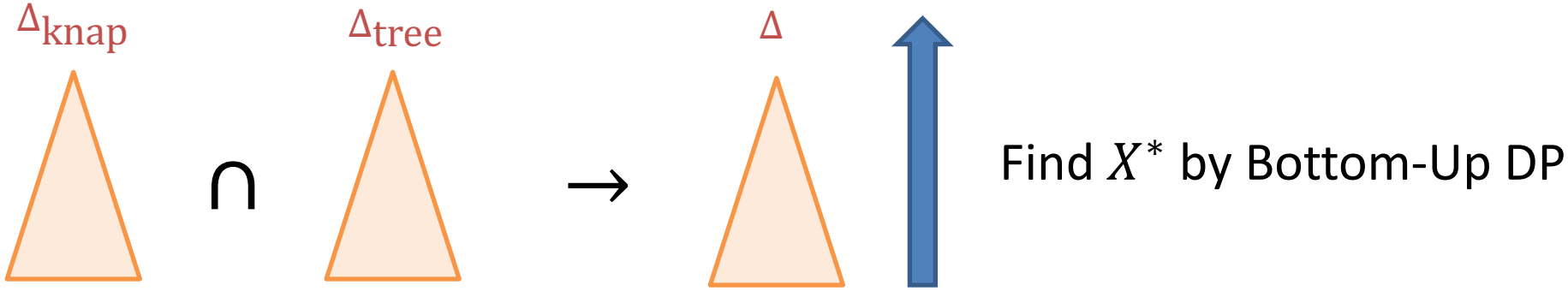
- $X^* \coloneqq \mathrm{argmax}_{X \in \mathcal{C}} \sum_{i \in X} w_i$: $O(|\Delta|)$ で計算可能
- $\mathcal{C} \coloneqq \mathcal{C}_{\mathrm{knap}} \cap \mathcal{C}_{\mathrm{tree}}$: ZDD の Operation で構成可能
- $\mathcal{C}_{\mathrm{knap}} \coloneqq \{X \subseteq [N] \mid \sum_{i \in X} \ell_i \leq L\}$: $|\Delta_{\mathrm{knap}}| = O(NL)$
- $\mathcal{C}_{\mathrm{tree}} \coloneqq \{X \subseteq [N] \mid T[X] \text{ is tree.}\}$: $|\Delta_{\mathrm{tree}}| = O(N)$





提案手法

1. Construct Δ_{knap}: the ZDD for $\mathcal{C}_{\mathrm{knap}}$ in Top-down manner
2. Construct Δ_{tree}: the ZDD for $\mathcal{C}_{\mathrm{tree}}$ by Frontier-based Search
3. Construct Δ: the ZDD for $\mathcal{C} = \mathcal{C}_{\mathrm{knap}} \cap \mathcal{C}_{\mathrm{tree}}$ by ZDD operation
4. Find $X^* := \mathrm{argmax}_{X \in \mathcal{C}} \sum_{i \in X} w_i$ on the ZDD Δ

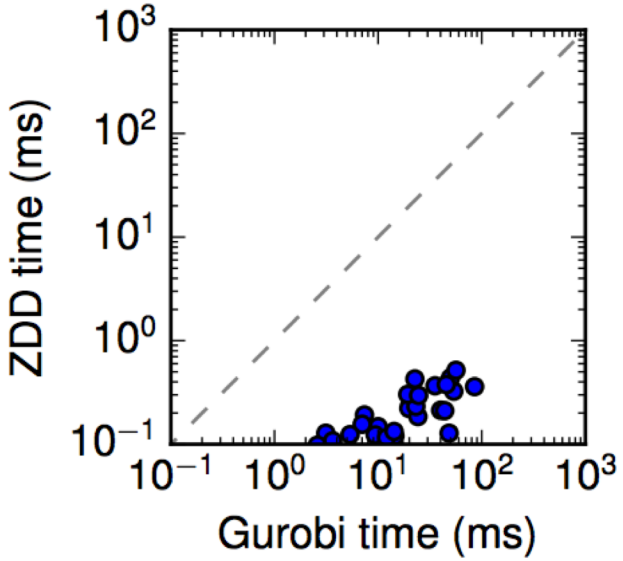


実験

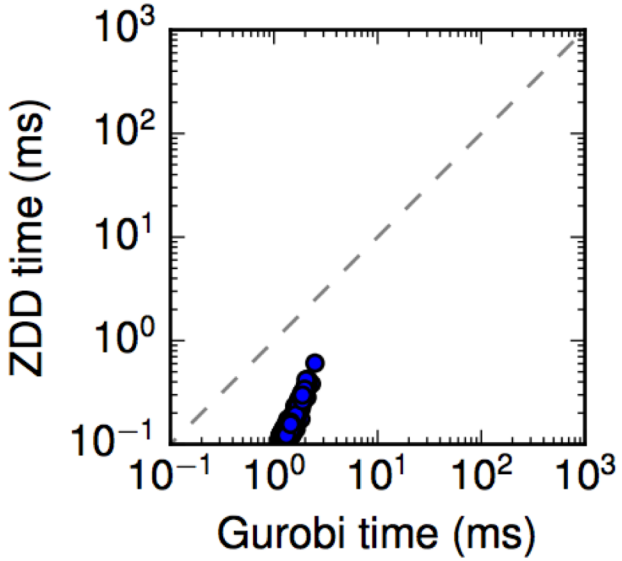
- 比較対象
  - 提案法
  - 既存法：整数計画問題としての定式化 + Grobi　※近似解法

- 比較内容
  1. 文書要約ベンチマークセットで計算時間を比較
  2. 木のサイズを大きくしたときの計算時間を比較

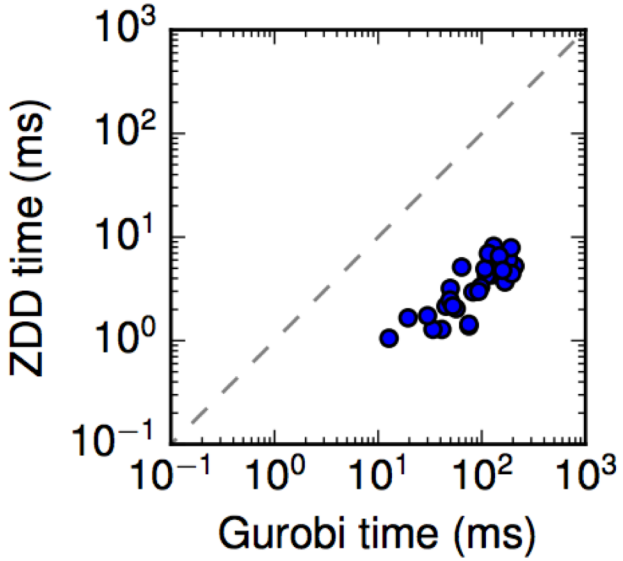
実験結果1：ベンチマークの速度比較



(a) Sentence extraction

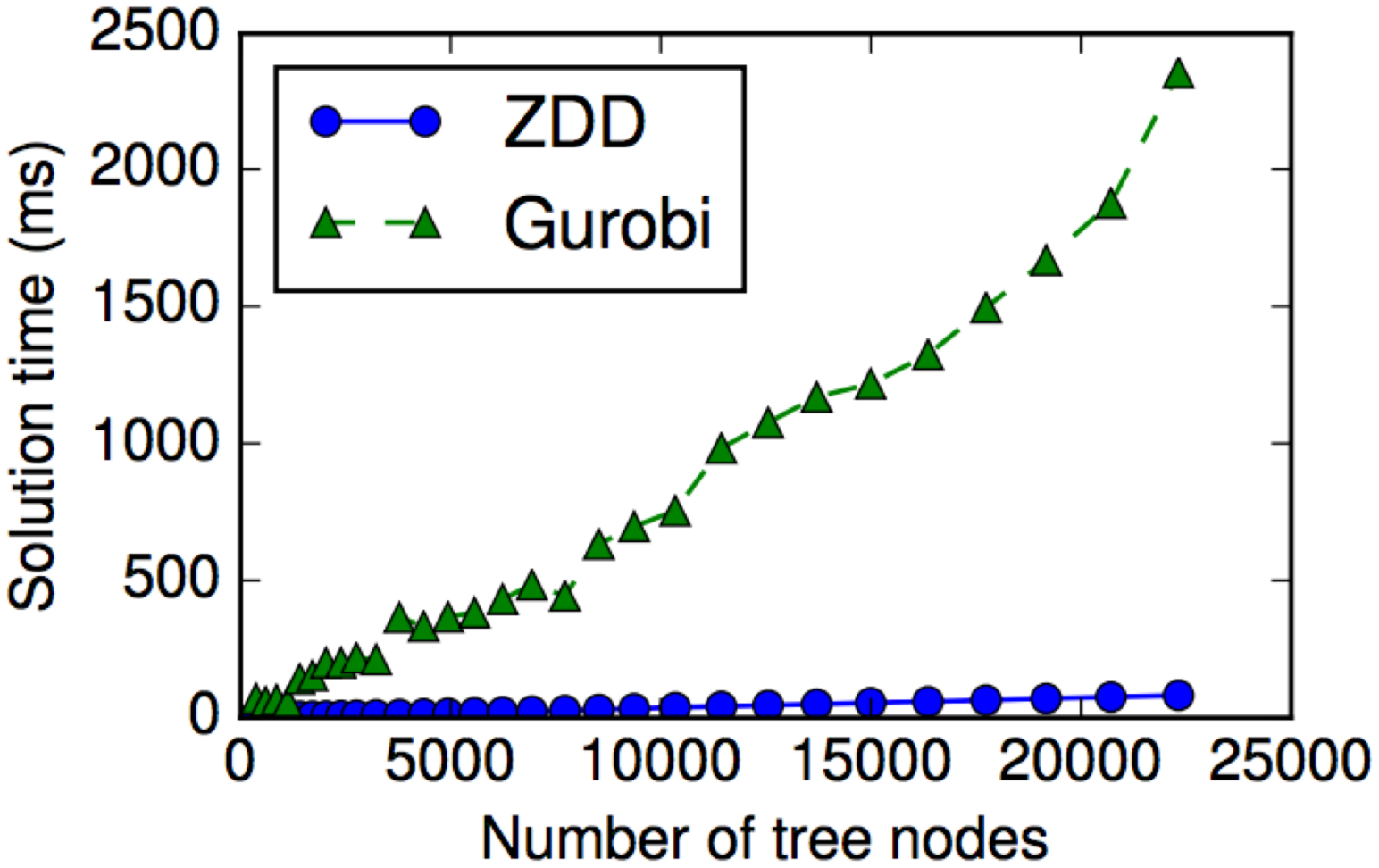


(b) Sentence compression



(c) Extraction & compression

実験結果2：木のノードの増加と計算速度



１枚概要

- 重要文抽出
  - 与えられた文書から重要な文を取り出して要約を作る
  - 素朴に定式化するとナップザック問題
- 木刈り込みによる要約
  - 各文の依存関係を表す木が与えられ、木制約のもとで要約を作成
  - 厳密最大化手法が知られていない
- ZDDを用いた効率的な木刈り込み法を提案
  - ZDDで木制約＋ナップサック制約を表現
  - 既存手法より数百倍高速

[2/4]
Exact Computation of Influence Spread by Binary Decision Diagrams

Maehara, Suzuki, Ishihata

WWW'2017

1枚概要

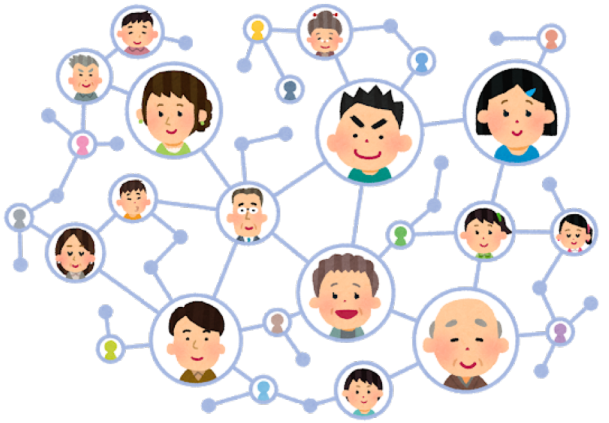
- 影響最大化問題
  - ネットワークのK頂点を選び、影響拡散を最大化する問題
  - 影響拡散の計算はNP困難
- BDDを用いた影響拡散の厳密計算法を提案
  - BDDで情報伝達過程を表す部分グラフ全体を表現
  - BDD上で情報伝達確率を計算
- 提案法を実ネットワークに適用
  - 数百頂点のグラフで影響拡散を厳密に計算
  - 影響力最大化も実行

影響最大化問題

- 影響最大化問題 [Kemp+ KDD'03]

  与えられたネットワークにおいて、

  影響拡散 $\sigma(S)$ が最大となる頂点集合 S ($|S| = K$)を選択する問題



- 影響拡散 $\sigma(S)$

  頂点集合 S から情報が到達する頂点数の期待値

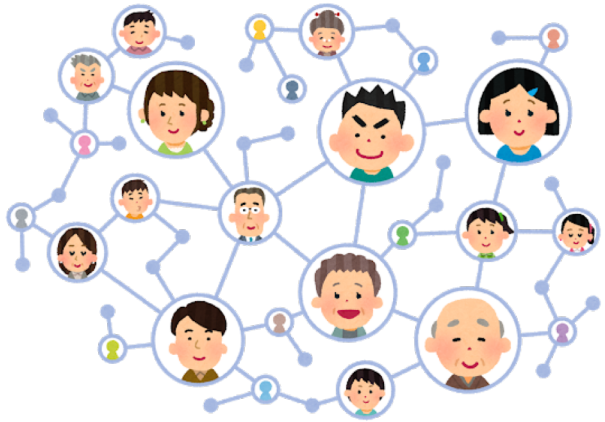
  ※ 情報はある情報拡散モデルに従って確率的に伝播(後述)

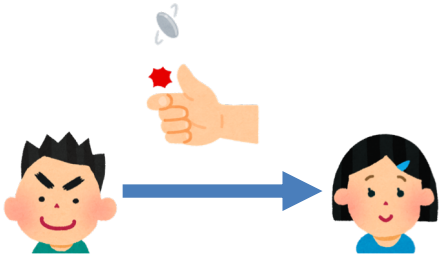
- 影響最大化の応用

  バイラルマーケティング

影響最大化問題

- Input:
  - 有向グラフ $G := \langle V, E \rangle$
  - 情報伝播確率 $p: E \to [0,1]$
  - 定数 K





$p(e)$ で情報が伝達

- Output:
  - $S^* := \mathrm{argmax}_{S \in \mathcal{C}_K} \sigma(S)$
  - $\mathcal{C}_K := \{S \subseteq V \mid |S| = K\}$

☹ 一般的にはNP困難

☹ $\sigma(S)$ は非線形なのでBDDでは解けない

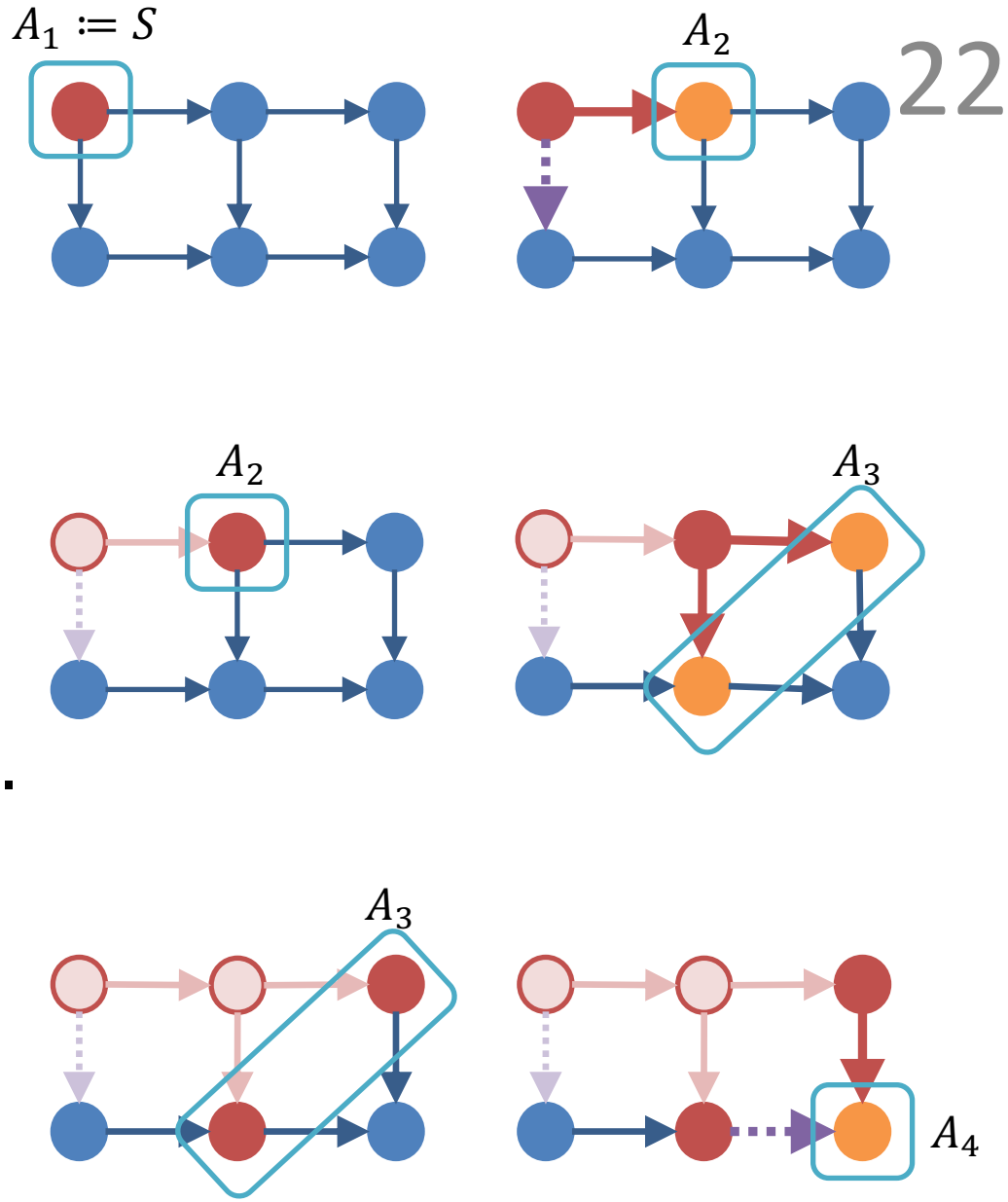
☺ $\sigma(S)$ は劣モジュラ関数

→ 貪欲法で 1-1/e 近似可能

カスケードモデル

1. Initialization: Let $A_1 := S$.
2. Spread: For $t = 1,2, \ldots$
   1. Terminate if $|A_t| = 0$.
   2. Activate all $v \in A_t$ and set $A_{t+1} = \emptyset$.
   3. For each $e = (u, v) \in E : u \in A_t$,

      add v to A_{t+1} with probability $p(e)$.



カスケードモデルにおける影響拡散

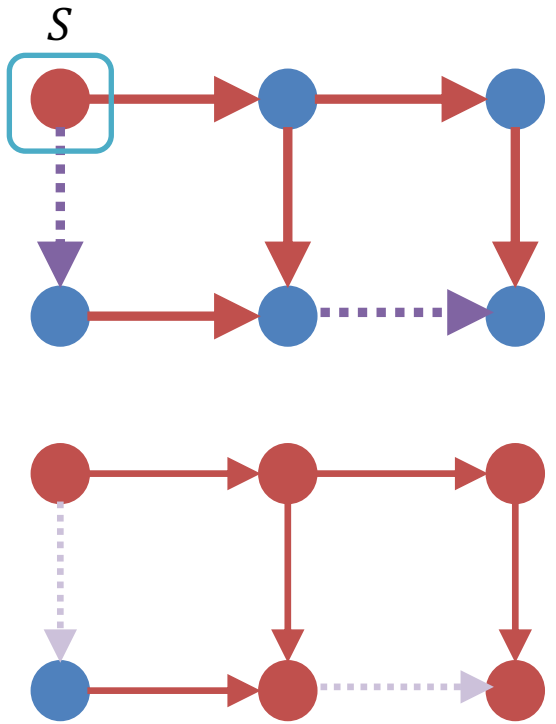
- 情報伝達：$S \to v$ =「S から v に情報が伝わる」

  ⇔ 各辺 $e \in E$ を確率 $p(e)$ でサンプリングして $X \subseteq E$ を得たとき、

  部分グラフ $G[X]$ において S のいずれの頂点から v に有向パスが存在 $(S \to_{G[X]} v)$

- 情報伝達確率：$p(S \to v) \coloneqq \sum_{X \in \mathcal{R}(S,v)} p(X)$
  - $p(X) \coloneqq \prod_{e \in X} p(e) \prod_{e \notin X} (1 - p(e))$
  - $\mathcal{R}(S, v) \coloneqq \{X \subseteq E \mid S \to_{G[X]} v\}$

- 影響拡散：$\sigma(S) = \sum_{v \in V} p(S \to v)$



$X \sim p(X)$

$S \to_{G[X]} v$

本研究

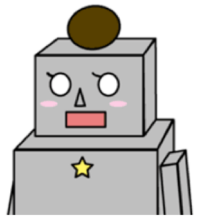
- 影響拡散 $\sigma(S)$ の効率的な厳密計算法を提案
  - BDD で情報伝達 $S \to v$ を表す部分グラフ集合 $\mathcal{R}(S, v)$ を表現
  - BDD 上の DP で情報伝達確率 $p(S \to v)$ を計算

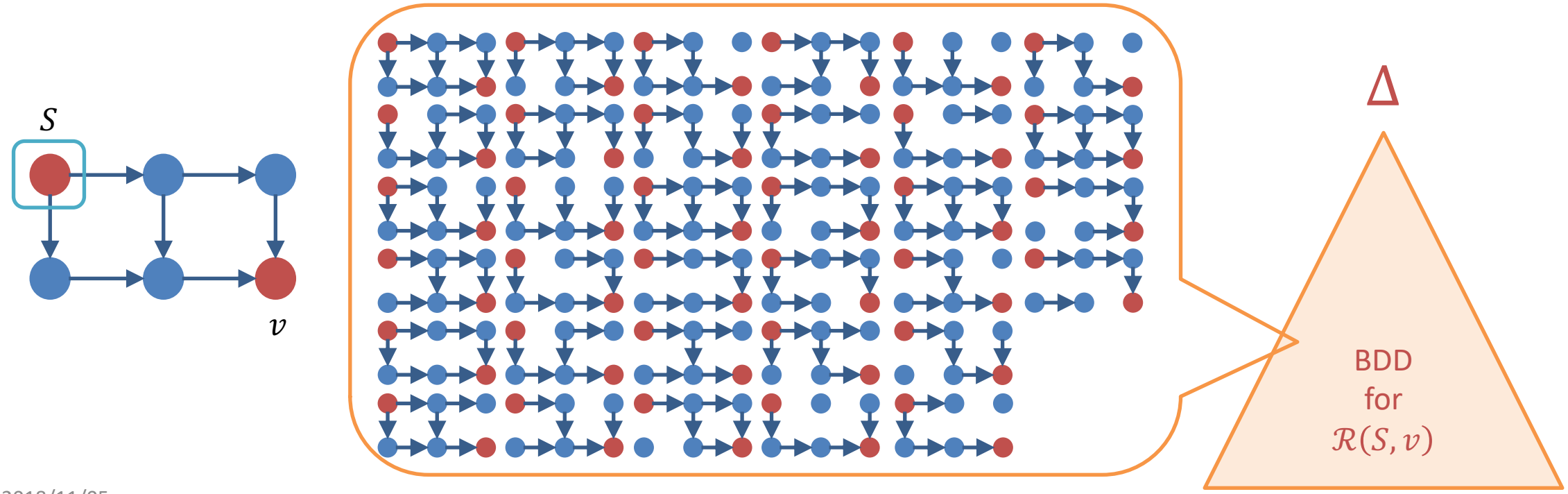
- 提案手法＋貪欲法で実ネットワークの影響最大化を実行
  - 頂点数が数百のグラフにおいて影響拡散を厳密評価
  - サンプリングによる情報拡散の近似精度も評価

今回のDDポイント：

$\mathcal{R}(S,v) \subseteq 2^E$ なので一般的には指数サイズ

- $\sigma(S) \coloneqq \sum_{v \in V} p(S \to v)$
- $p(S \to v) \coloneqq \sum_{X \in \mathcal{R}(S,v)} p(X)$ ：$O(|\Delta|)$ で計算可能
- $\mathcal{R}(S,v) \coloneqq \{X \subseteq E : S \to_{G[X]} v\}$ ：フロンティア法でBDDを構成可能





提案法：Greedy Search using BDDs

1. Initialization: find the best single node
    1. For each $u, v \in V$, construct the BDD for $\mathcal{R}(\{u\}, v)$ by Frontier-based Search and evaluate $p(\{u\} \to v)$ on the BDD.
    2. Find $u^* = \mathrm{argmax}_{u \in V} \sigma(\{u\})$, and let $S \coloneqq \{u^*\}$.

2. Repeat until $|S| = K$: add nodes by greedy
    1. For each $u, v \in V$, construct the BDD for $\mathcal{R}(S \cup \{u\}, v)$ by BDD operations and evaluate $p(S \to v)$ on the BDD.
    2. Find $u^* = \mathrm{argmax}_{u \in V} \sigma(S \cup \{u\})$, and let $S \coloneqq S \cup \{u^*\}$.

実験：グラフサイズとBDDサイズ

Table 1: Computational results on real-world networks. Time denotes the average time to construct the BDDs, BDD Size denotes the average number of nodes in the BDDs, Shared Size denotes the total number of distinct nodes in the shared BDDs, and Cardinality denotes the average number of subgraphs represented by the BDDs. Here, average is taken of all distinct $s, t \in V$. For the last three networks, the algorithm failed to compute due to the memory limit.

Network	Vertices	Edges	Time [ms]	BDD Size	Shared Size	Cardinality
South-African-Companies	11	26	0.1	12.1	472	2.2e+07
Southern-women-2	20	28	0.3	54.7	2,266	1.3e+08
Taro-exchange	22	78	4.1	1,119.2	277,756	1.6e+23
Zachary-karate-club	34	156	24.9	7,321.8	4,988,148	6.4e+46
Contiguous-USA	49	214	117.9	30,599.8	41,261,047	1.6e+64
American-Revolution	141	320	2.2	120.0	1,530,677	5.7e+95
Southern-women-1	50	178	—	—	—	—
Club-membership	65	190	—	—	—	—
Corporate-Leadership	64	198	—	—	—	—

実験：人工グラフのBDDサイズ

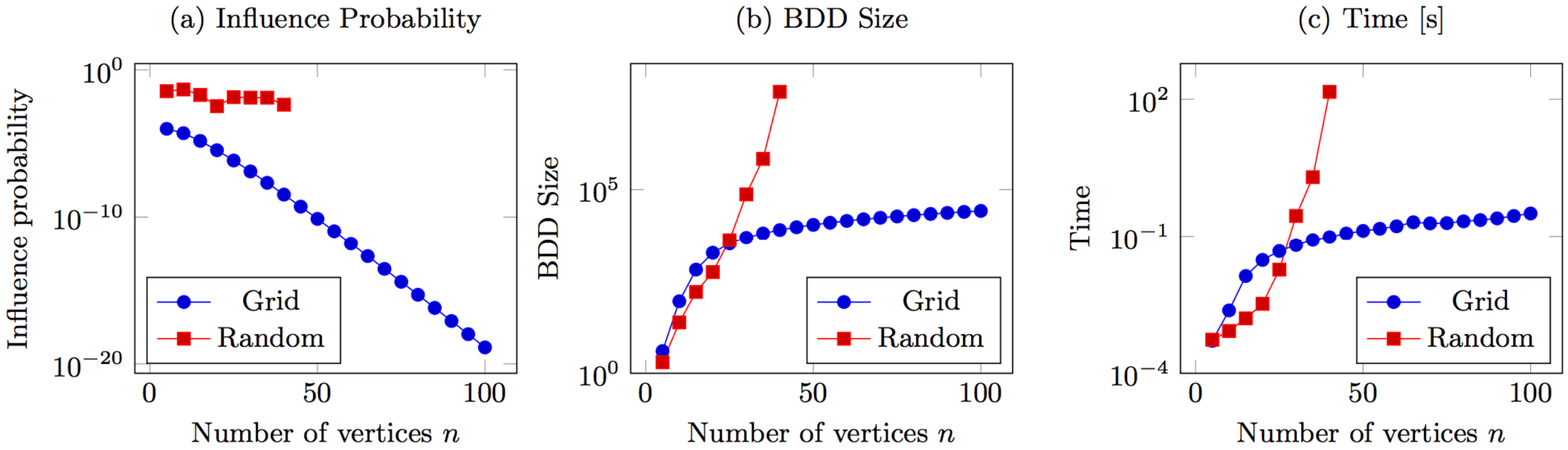


Figure 3: Computational results on $5 \times w$ grid graphs and random graphs. The algorithm failed to compute the influence spread on the random network with $n \geq 45$ vertices due to the memory limit.

実験：実ネットワークで影響最大化

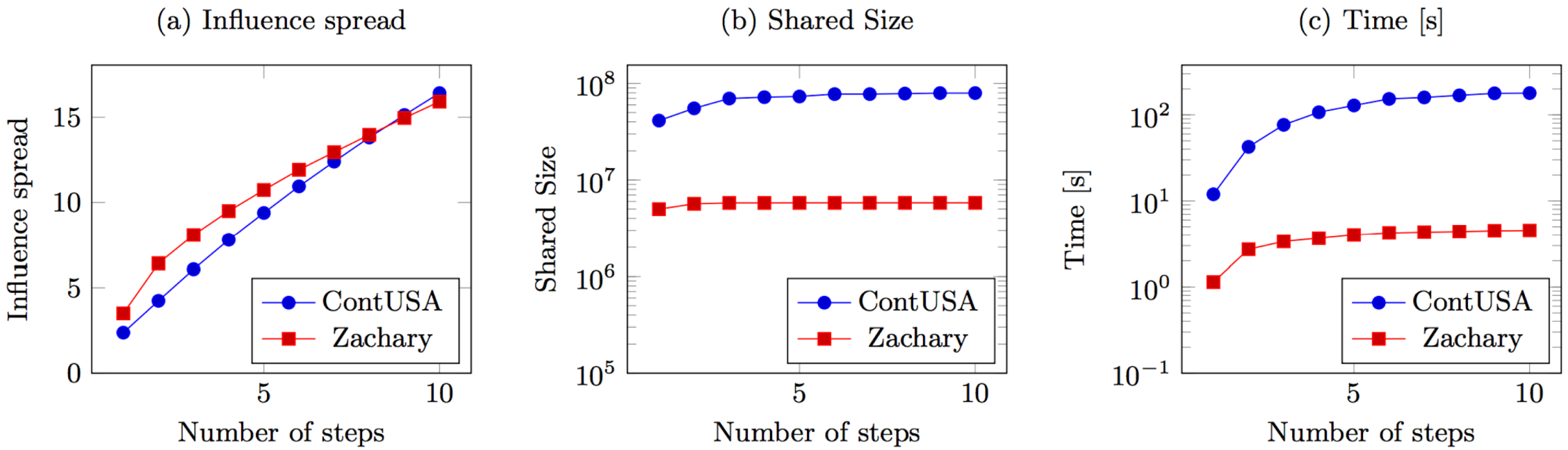


Figure 4: Computational results on the influence maximization problem with exact influence spread.

実験:サンプル近似の誤差評価

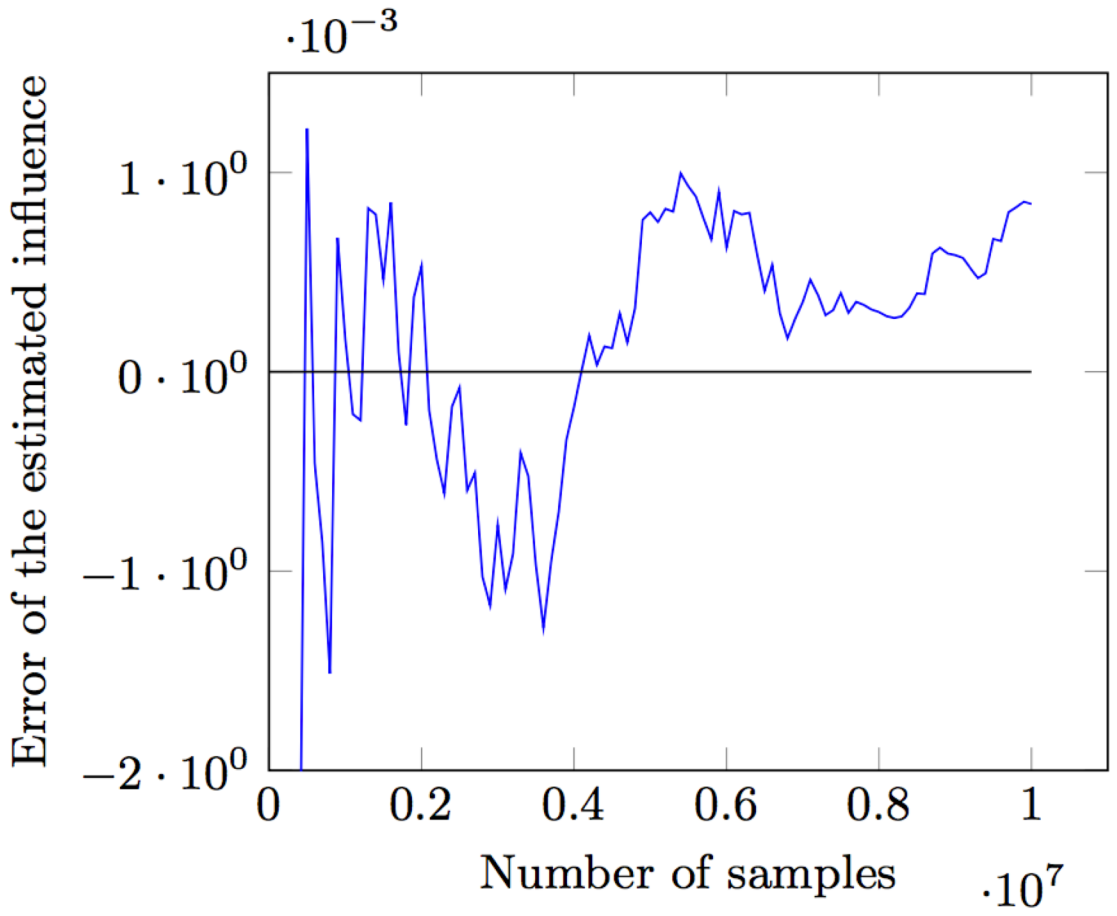


Figure 5: Accuracy of Monte-Carlo simulation.

１枚概要

- 影響最大化問題
  - ネットワークのK頂点を選び、影響拡散を最大化する問題
  - 影響拡散の計算はNP困難
- BDDを用いた影響拡散の厳密計算法を提案
  - BDDで情報伝達過程を表す部分グラフ全体を表現
  - BDD上で情報伝達確率を計算
- 提案法を実ネットワークに適用
  - 数百頂点のグラフで影響拡散を厳密に計算
  - 影響力最大化も実行

[3/4]
Efficient Bandit Combinatorial Optimization Algorithm with Zero-suppressed Binary Decision Diagrams

Sakaue, Ishihata, Minato

AISTATS'2018

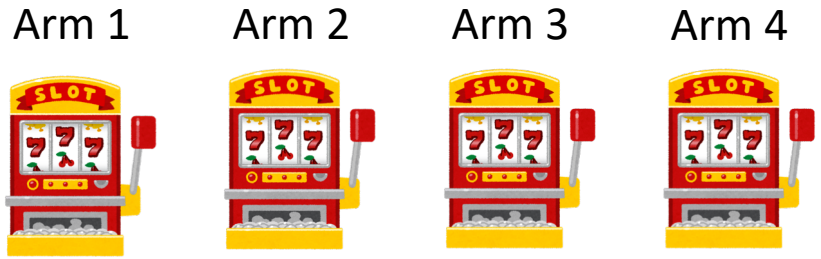
１枚概要

- 敵対的組合せバンディット問題
  - オンライン最適化問題の一種
  - 与えられた組合せの集合から良さそうなものを選ぶ問題

- 既存のアルゴリズムは非実用的な計算量
  - 理論的なバウンドが議論の対象
  - 指数的な計算時間がかかっても問題なし

- ZDDを用いた敵対的組合せバンディット問題に対する実用的なアルゴリズムを提案
  - ZDDで対象の組合せ集合を表現
  - ZDD上でオンライン最適化を実行（圧縮したまま計算できる）

バンディット問題



- K 本の arm（スロットマシン）$i \in [K]$ が存在
- 各時刻 $t = 1, \ldots, T$ で arm $i \in [K]$ を1つ選択
  - $r_t(i) :=$ 時刻 t に arm i から得られる利得
  - $i_t :=$ 時刻 t で選んだ arm
  - 観測できるのは $r_t(i_t)$ のみ

- 以下のリグレット（後悔）が小さくなるように arm を選びたい

$$R_T := \underbrace{\max_{i \in [K]} \sum_{t \in [T]} r_t(i)}_{①} - \underbrace{\sum_{t \in [T]} r_t(i_t)}_{②}$$

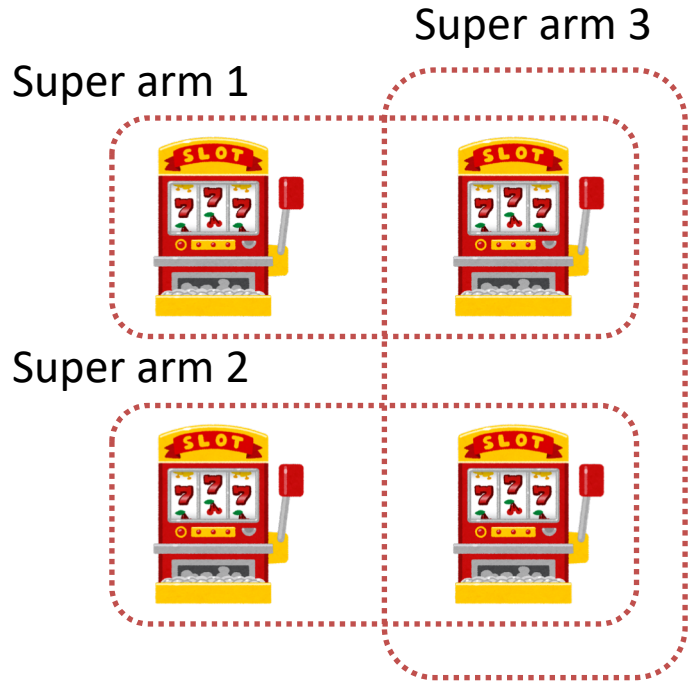
① Best Single Arm の総利得

② 実際の総利得

敵対的バンディット問題

- 非敵対的バンディット問題
  - 利得 $r_t(i)$ が t に独立な分布から得られる
  - 例: $r_t(i) \sim \mathrm{Bernoulli}(\mu_i)$
- 敵対的バンディット問題
  - 利得 $r_t(i)$ が t に依存した分布から得られる
  - 過去の選択 $\{ i_{t'} \mid t' \in [t-1] \}$ に依存してもよい
  - 例: $r_t(i) \sim \mathrm{Bernoulli}(\mu_{i,t})$, $\mu_{i,t} \coloneqq |\{ t' \in [t-1] \mid i_{t'} = i \}|^{-1}$

組合せバンディット問題



- K 本の arm $i \in [K]$ が存在
- $\mathcal{S} \subseteq 2^{[K]}$ なる super arm $X \in \mathcal{S}$が存在

- 各時刻 $t = 1, \ldots, T$ で super arm $X \in \mathcal{S}$ を1つを選択
  - $r_t(i) :=$ 時刻 t に arm $i \in [K]$ から得られる利得
  - $r_t(X) := \sum_{i \in X} r_t(i)$: 時刻 t に super arm $X \in \mathcal{S}$ から得られる利得
  - $X_t :=$ 時刻 t で選んだ super arm
  - 観測できるのは $r_t(X_t)$ のみ　注意: $\{ r_t(i) \mid i \in X_t \}$ は観測不可

- 以下のリグレット（後悔）が小さくなるように super arm を選びたい

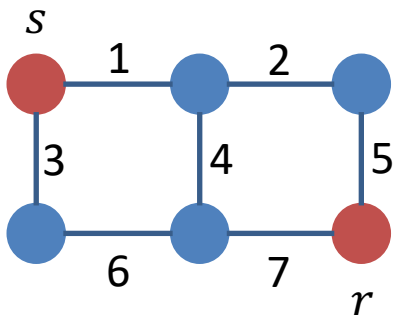
$$R_T := \underbrace{\max_{X \in \mathcal{S}} \sum_{t \in [T]} r_t(X)}_{①} - \underbrace{\sum_{t \in [T]} r_t(X_t)}_{②}$$

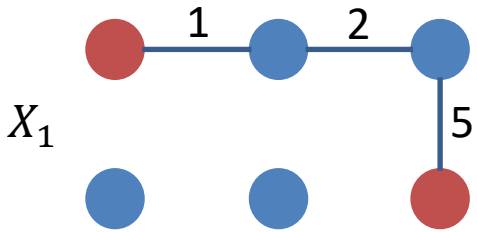
① Best Single Super Arm の総利得

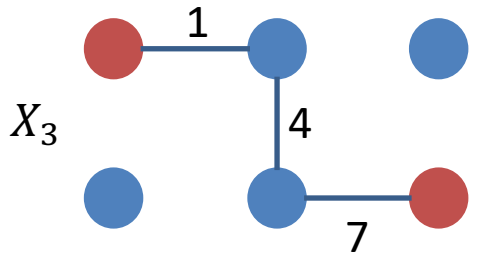
② 実際の総利得

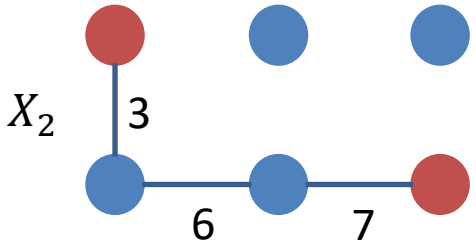
例：Online Shortest Path (OSP)

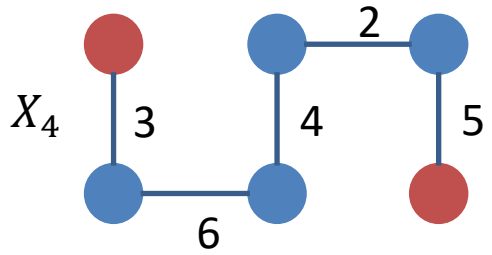
- 通信ネットワーク上の Sender s から Receiver r へ T message 送る
- 各辺の通信効率 $r_t(i)$ はネットワークの使用状況に応じて刻々と変化する
- すべてのメッセージを送る効率を最大化したい











$$K = 7$$

$$\mathcal{S} = \{X_1, X_2, X_3, X_4\}$$

$$X_1 = \{1,2,5\}$$

$$X_2 = \{3,6,7\}$$

$$X_3 = \{1,4,7\}$$

$$X_4 = \{2,3,4,5,6\}$$

既存手法 [Cesa-Bianchi+ 12]

1. Given $\mathcal{S}$ as list form
2. For each $i \in [K]$, initialize $w_1(i) = 1$
3. Repeat $t = 1, \ldots, T$:
   1. Sample super arm $X_t \sim p(X_t \mid \mathcal{S})$
   2. Observe reward $r_t(X_t) = \sum_{i \in X_t} r_t(i)$
   3. Compute $p_{i,j} := p(i, j \in X_t \mid \mathcal{S})$
   4. Estimate $\hat{r}_t(i)$ using $\mathrm{P} := \{p_{i,j}\}_{i,j}$
   5. Update $w_{t+1}(i) := w_t(i)\exp(-\eta \hat{r}_t(i))$

$$p(X_t \mid \mathcal{S}) = \frac{w_t(X)}{\sum_{X \in \mathcal{S}} w_t(X)}$$

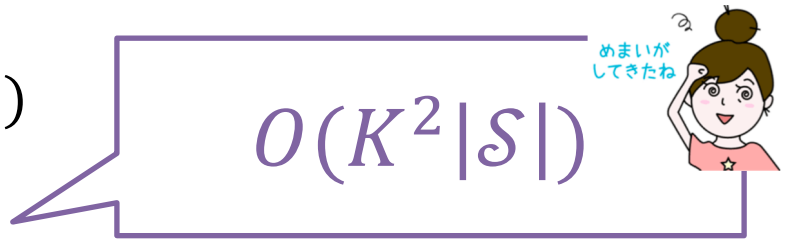
$$w_t(X) := \prod_{i \in X} w_t(i)$$

Arm i の有望さ

Super Arm X の有望さ

既存手法 [Cesa-Bianchi+ 12]

1. Given $\mathcal{S}$ as list form
2. For each $i \in [K]$, initialize $w_1(i) = 1$.
3. Repeat $t = 1, \ldots, T$:
   1. Sample super arm $X_t \sim p(X_t \mid \mathcal{S})$
   2. Observe reward $r_t(X_t) = \sum_{i \in X_t} r_t(i)$
   3. Compute $p_{i,j} \coloneqq p(i, j \in X_t \mid \mathcal{S})$
   4. Estimate $\hat{r}_t(i)$ using $\mathrm{P} \coloneqq \{p_{i,j}\}_{i,j}$
   5. Update $w_{t+1}(i) \coloneqq w_t(i)\exp(-\eta \hat{r}_t(i))$

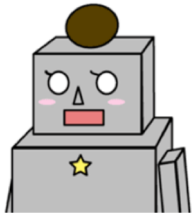


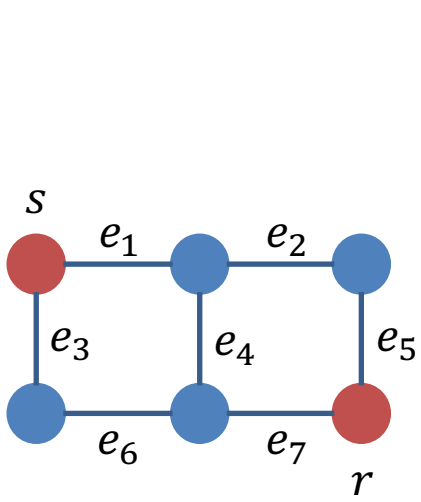
本研究

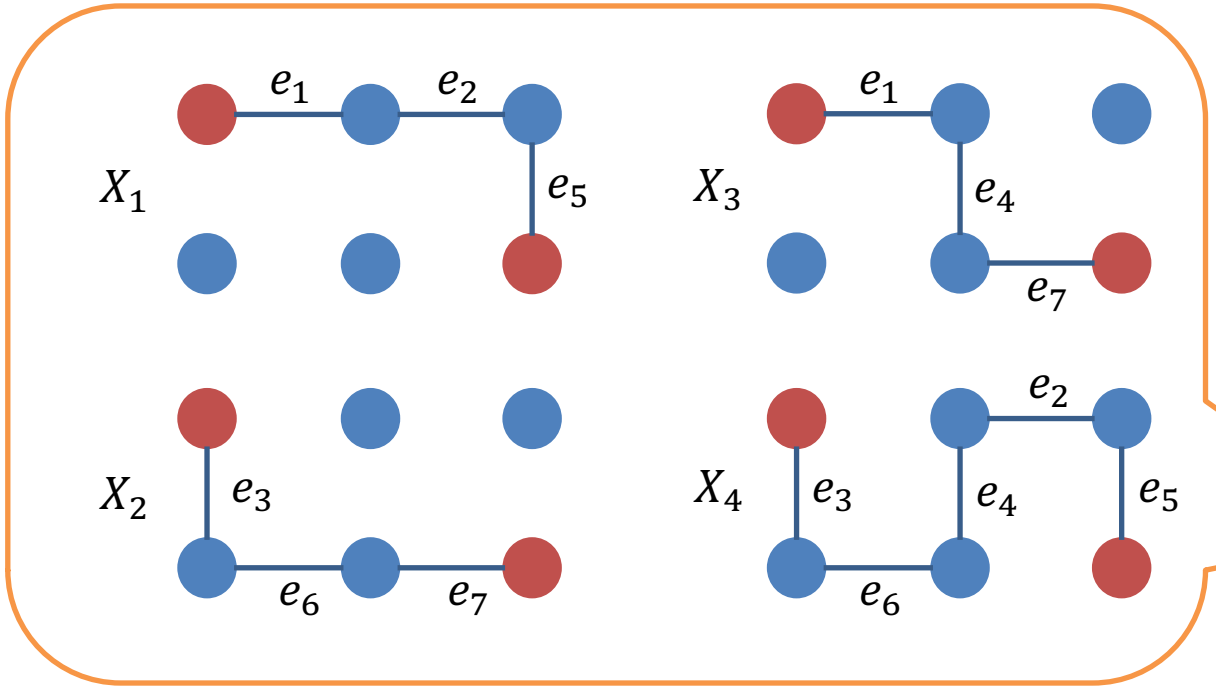
- 敵対的組合せバンディット問題に対する効率的な手法を提案
  - ZDD で super arm 集合 S を表現
  - ZDD を用いてサンプリングと共起確率を計算
- 実験的に既存法より数百倍高速であることを確認
  - ３種類の問題を考えて、それぞれで実験して Regret を確認
  - そもそもこの問題設定で実験をした論文はほぼない

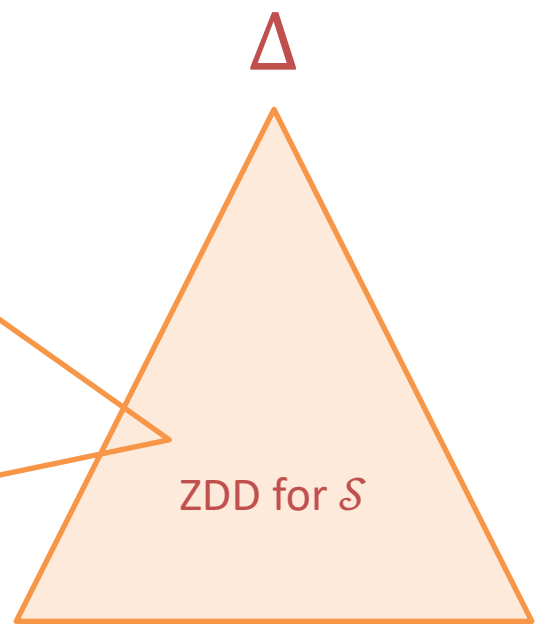
今回のDDポイント：$\mathcal{S} \subseteq 2^{[K]}$ なので一般的に $|\mathcal{S}|$ は指数サイズ

- Sample $X_t \sim p(X_t \mid \mathcal{S})$ ：$O(|\Delta|)$ で計算可能
- Compute $p_{i,j} := p(i,j \in X_t \mid \mathcal{S})$ ：$O(K|\Delta|)$で計算可能









提案手法

1. Given $\mathcal{S}$ as ZDD D
2. For each $i \in [K]$, initialize $w_1(i) = 1$.
3. Repeat $t = 1, \ldots, T$:
   1. Sample super arm $X_t \sim p(X_t \mid \mathcal{S})$ on ZDD D
   2. Observe reward $r_t(X_t) = \sum_{i \in X_t} r_t(i)$
   3. Compute $p_{i,j} \coloneqq p(i, j \in X_t \mid \mathcal{S})$ on ZDD D
   4. Estimate $\hat{r}_t(i)$ using $\mathrm{P} \coloneqq \{p_{i,j}\}_{i,j}$
   5. Update $w_{t+1}(i) \coloneqq w_t(i)\exp(-\eta \hat{r}_t(i))$

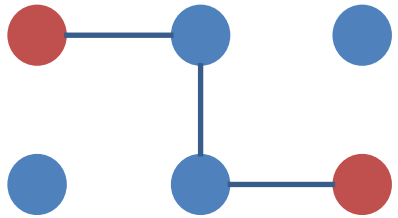
$O(K|\mathcal{S}|) \rightarrow O(|\Delta|)$

$O(K^2|\mathcal{S}|) \rightarrow O(K|\Delta|)$

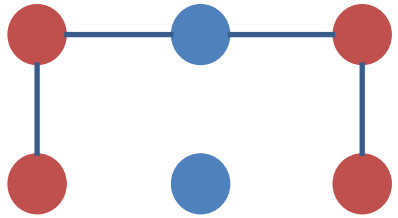
※本当はリグレットも改善したけどDDと関係ないので詳細は省略

実験

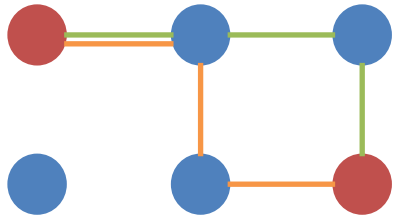
- Online Shortest Path (OSP) on Grid
  - プレイヤーが Grid の左上から右下に至る Path を選択する問題
  - 辺の利得は過去の履歴に応じて変化



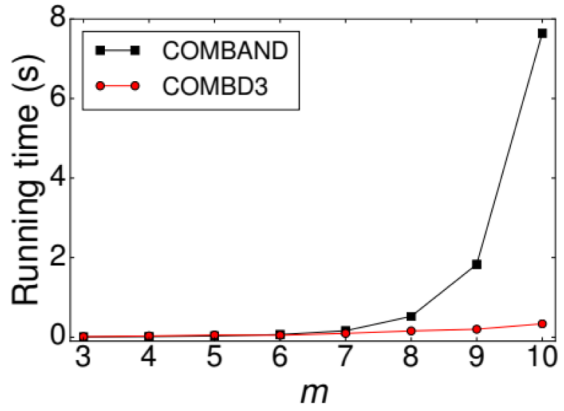
- Dynamic Steiner Tree (DST) on Grid
  - プレイヤーが Grid の四隅を含む Tree を選択する問題
  - 辺の利得は過去の履歴に応じて変化



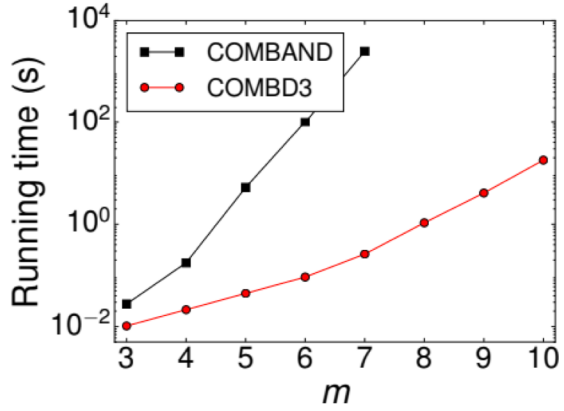
- Congestion Game on Small Real-world Networks
  - ２人のプレイヤーが指定した２点間の Path を選択する問題
  - 辺の利得は相手の選択に応じて変化



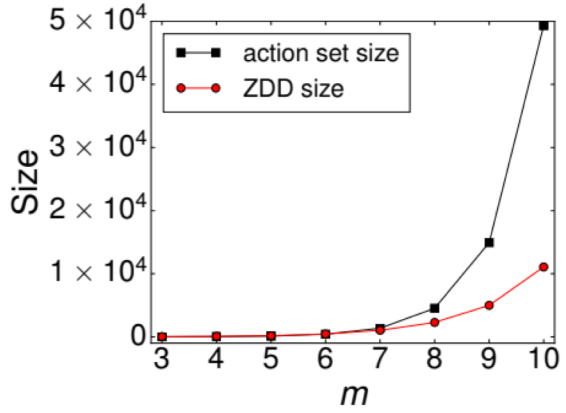
実験結果：OSP & DST



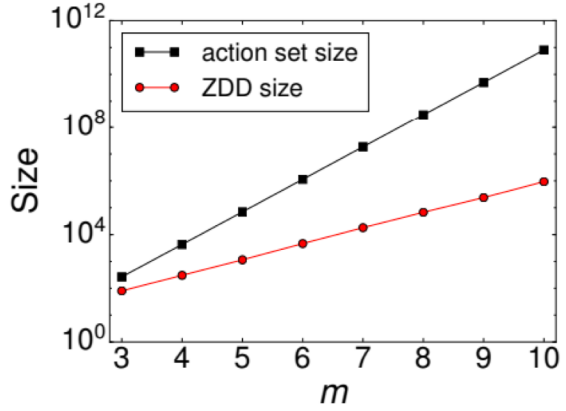
(a) OSP



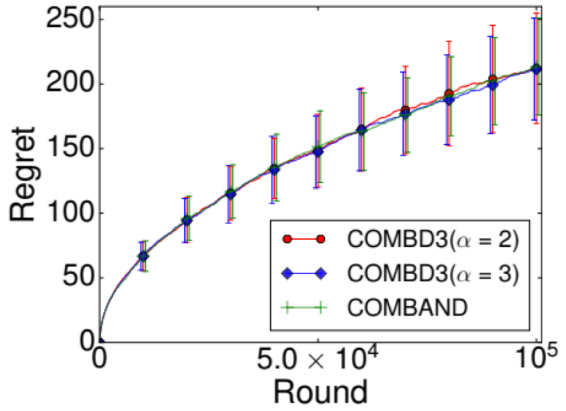
(b) DST (semi-log)



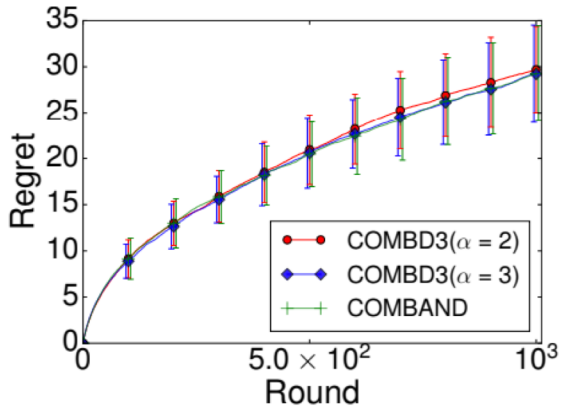
(c) OSP



(d) DST (semi-log)

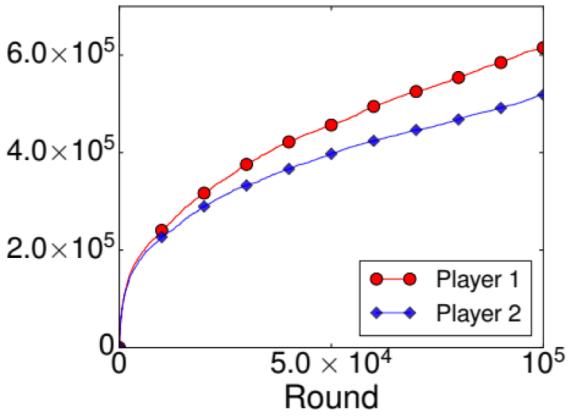


(e) OSP

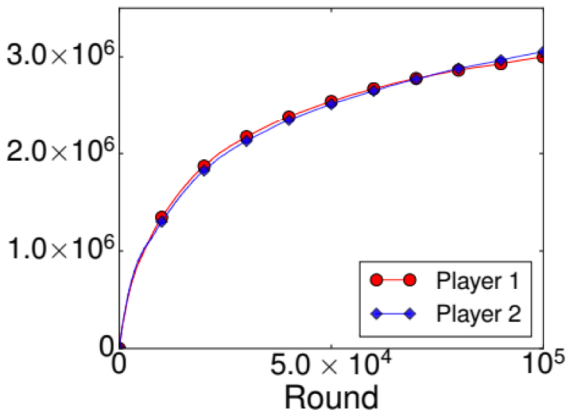


(f) DST

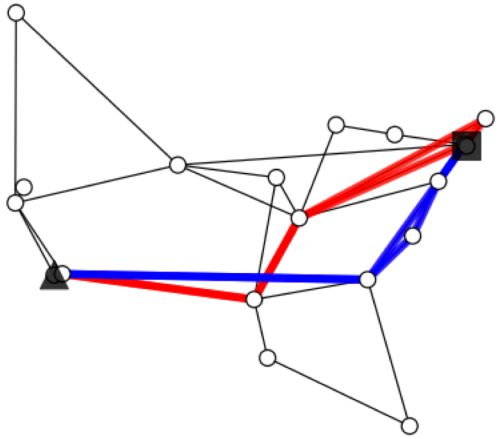
実験結果：CG



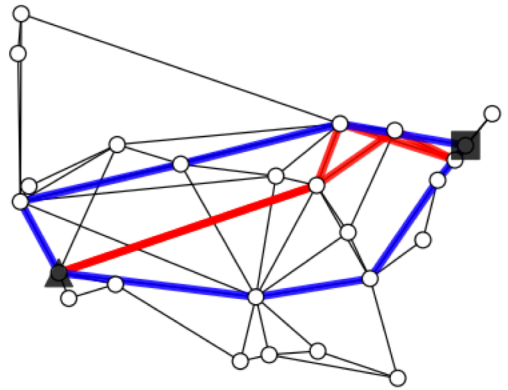
(g) CG on MCI



(h) CG on ATT



(a) MCI



(b) ATT

１枚概要

- 敵対的組合せバンディット問題
  - オンライン最適化問題の一種
  - 与えられた組合せの集合から良さそうなものを選ぶ問題

- 既存のアルゴリズムは非実用的な計算量
  - 理論的なバウンドが議論の対象
  - 指数的な計算時間がかかっても問題なし

- ZDDを用いた敵対的組合せバンディット問題に対する実用的なアルゴリズムを提案
  - ZDDで対象の組合せを表現
  - ZDD上でオンライン最適化を実行（圧縮したまま計算できる）

[4/4]
Optimizing Network Reliability via Best-First Search over Decision Diagrams

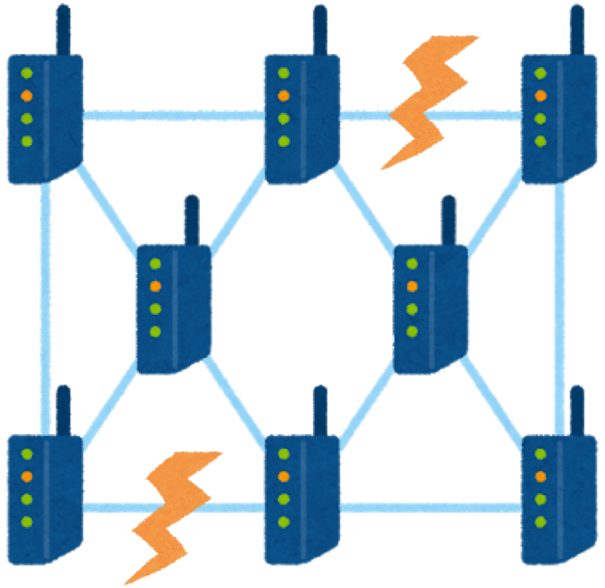
Nishino, Inoue, Yasuda, Minato, Nagata

INFOCOM’2018

1枚概要

- ネットワーク信頼性
  - 通信ネットワークの各辺が確率的に故障するとき、通信が保たれる（連結である）確率
  - ネットワーク信頼性の計算はNP困難
- 信頼性最大化問題
  - 限られた予算・設備内で、ネットワーク信頼性が最大の構成を見つける問題
  - ネットワーク信頼性が O(1) で計算できたとしてもNP困難
- BDD を用いた信頼性最大化問題に対する厳密アルゴリズムを提案
  - 最良優先探索により最適な構成を発見
  - ネットワーク信頼性と優先度の評価にBDDを利用

ネットワーク信頼性



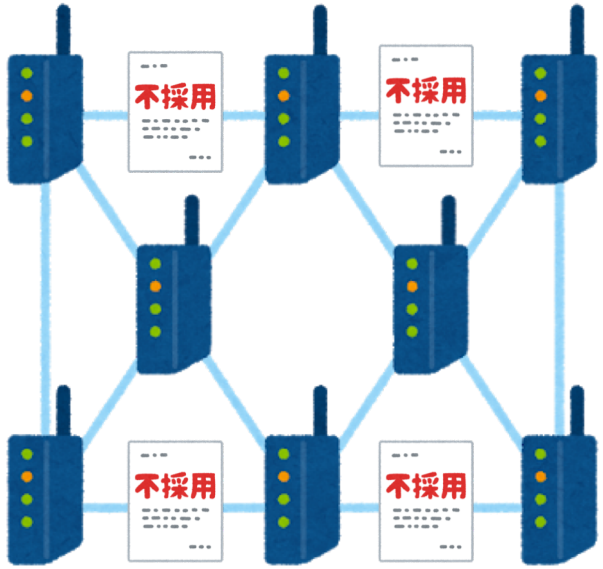
- 通信ネットワーク: $\langle G, p \rangle$
  - 無向グラフ $G := \langle V, E \rangle$
  - 辺信頼確率 $p : E \to [0,1]$

☹ 一般的には NP 困難

☺ BDD で効率的に計算できる

- ネットワーク信頼性 $R_p(G)$
  - $R_p(G) := E_p[G[X] \text{ is connected}] = \sum_{X \in \mathcal{C}_G} p(X)$
  - $\mathcal{C}_G := \{X \subseteq E \mid G[X] \text{ is connected}\}$

信頼性最大化問題



- Input:
  - 資源グラフ $G \coloneqq \langle V, E \rangle$
  - 辺信頼確率 $p : E \to [0,1]$
  - 利用可能辺数 K
- Output:
  - $X^* \coloneqq \mathrm{argmax}_{X \in \mathcal{C}_K} R_p(G[X])$
  - $\mathcal{C}_K \coloneqq \{X \subseteq E \mid |X| = K\}$

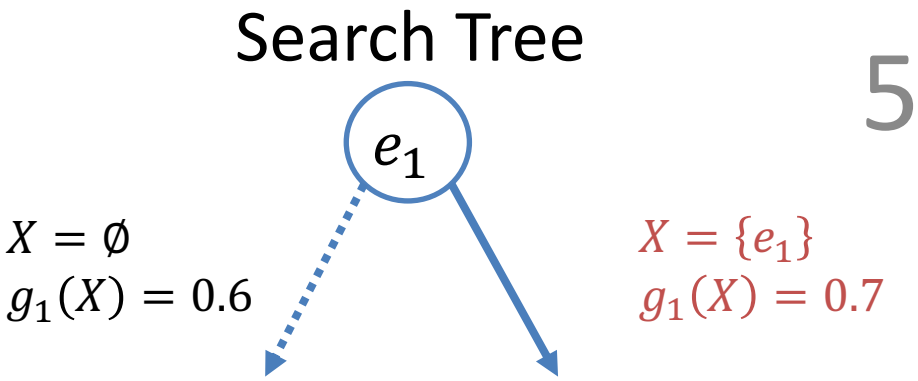
☹ 一般的には NP 困難

☹ R_p は非線形なのでBDDでは解けない

☺ R_p は単調増加

→ 最良優先探索ができる

最良優先探索



- 目的: $X^* := \mathrm{argmax}_{X \in \mathcal{C}_K} R_p(G[X])$
- 探索木: $e_i \in E$ を頂点として持つ完全二分木
- 優先度: 以下の2つの性質を満たす $g_i(X)$
  1. $g_i(X) \geq g^*(X)$: 必ず自分以下の解を過大評価する
  2. $g_i(X) = R_p(G[X])$ if $|X| = K$: 必ず最終的には正しい値と一致する

$g^*(X) := \max_{Y \in \mathcal{C}^{\geq i}_{K-|Y_i|}} R_P(G[X \cup Y])$: Y_i 以下で達成できる真の最大値

$\mathcal{C}^{\geq i}_L := \{Y \subseteq \{e_i, \ldots, e_N\} \mid |Y| = L\}$

- 最良優先探索
  - $g_i(X)$ が大きい探索頂点から展開する
  - 初めて見つかった $|X| = K$ なる候補は、$X = X^*$ を満たす

最良優先探索

- 目的: $X^* := \mathrm{argmax}_{X \in \mathcal{C}_K} R_p(G[X])$
- 探索木: $e_i \in E$ を頂点として持つ完全二分木
- 優先度: 以下の2つの性質を満たす $g_i(X)$

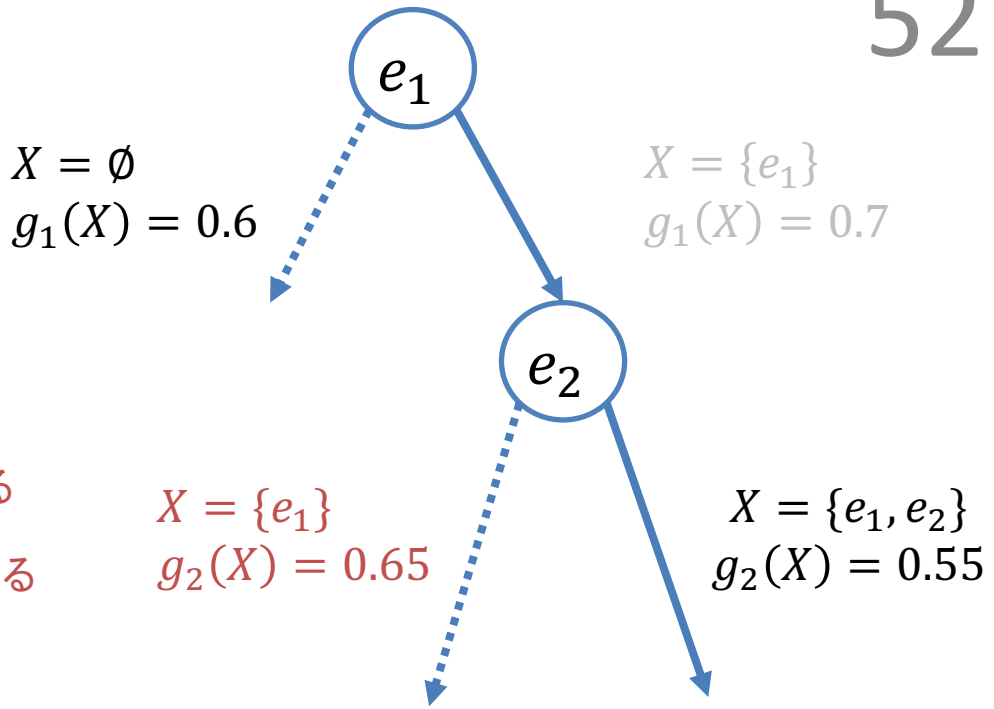
1. $g_i(X) \geq g^*(X)$: 必ず自分以下の解を過大評価する
2. $g_i(X) = R_p(G[X])$ if $|X| = K$: 必ず最終的には正しい値と一致する

$g^*(X) := \max_{Y \in \mathcal{C}^{\geq i}_{K-|Y_i|}} R_P(G[X \cup Y])$: Y_i 以下で達成できる真の最大値

$\mathcal{C}^{\geq i}_L := \{Y \subseteq \{e_i, \ldots, e_N\} \mid |Y| = L\}$

- 最良優先探索
  - $g_i(X)$ が大きい探索頂点から展開する
  - 初めて見つかった $|X| = K$ なる候補は、$X = X^*$ を満たす

Search Tree



最良優先探索

- 目的: $X^* := \mathrm{argmax}_{X \in \mathcal{C}_K} R_p(G[X])$
- 探索木: $e_i \in E$ を頂点として持つ完全二分木
- 優先度: 以下の2つの性質を満たす $g_i(X)$

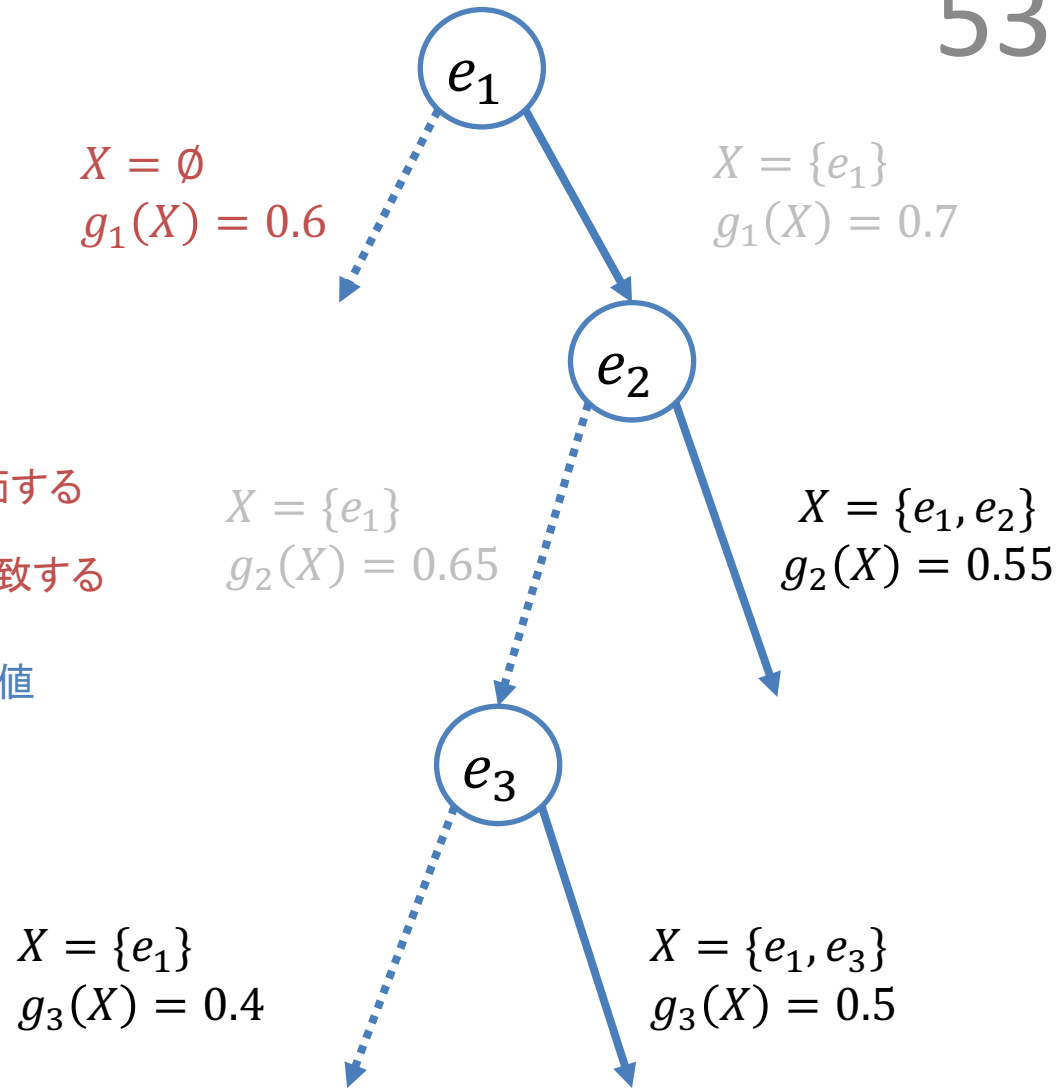
1. $g_i(X) \geq g^*(X)$: 必ず自分以下の解を過大評価する
2. $g_i(X) = R_p(G[X])$ if $|X| = K$: 必ず最終的には正しい値と一致する

$g^*(X) := \max_{Y \in \mathcal{C}^{\geq i}_{K-|Y_i|}} R_P(G[X \cup Y])$: Y_i 以下で達成できる真の最大値

$\mathcal{C}^{\geq i}_L := \{Y \subseteq \{e_i, \ldots, e_N\} \mid |Y| = L\}$

- 最良優先探索
  - $g_i(X)$ が大きい探索頂点から展開する
  - 初めて見つかった $|X| = K$ なる候補は、$X = X^*$ を満たす

Search Tree



本研究

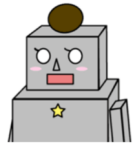
- （参考）ネットワーク信頼性の評価はBDDを用いて効率的に評価可能
  - 無向グラフ [Hardy+ 2007]
  - 有向グラフ [Suzuki-Ishihata-Minato COCOA’2018]

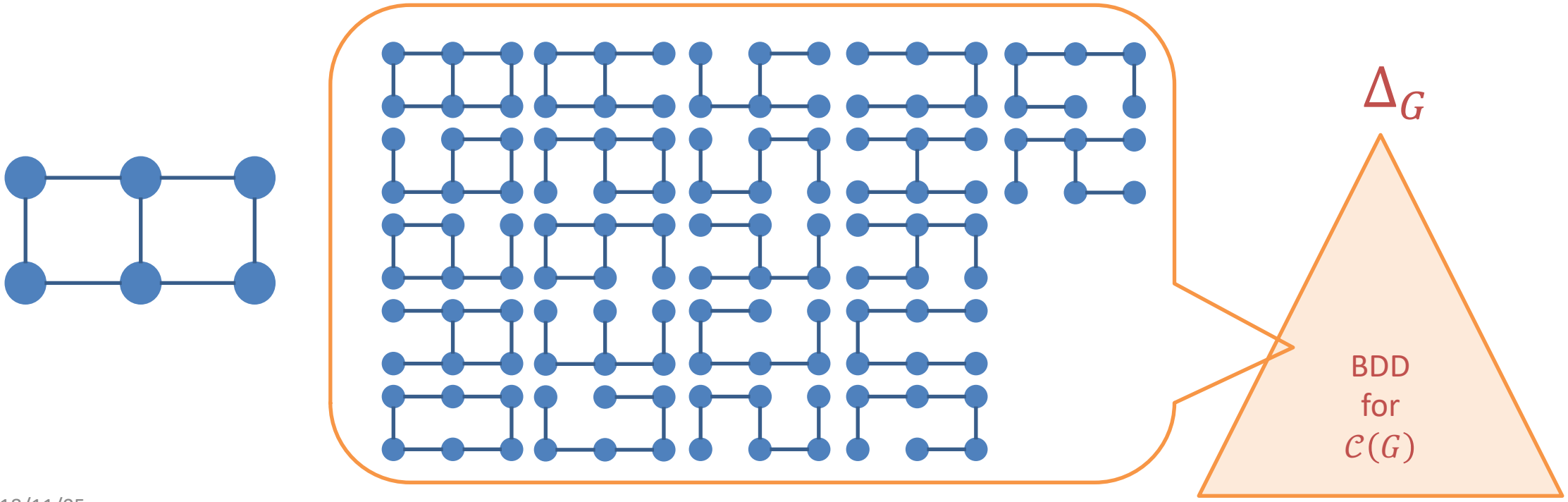
- 探索による信頼性最大化では膨大な数の構成候補の信頼性/優先度を評価
  - 全候補に対してBDDを作ることは非現実的
  - 信頼性だけでなく、優先度も BDD 上で効率的に評価したい

- 提案手法：単一のBDDですべての候補の信頼性を計算計算
  - 確率 p を変更することで単一のBDD上で信頼性評価を実現
  - 優先度も同様に評価も実現（割愛）

今回のDDポイント1：$\mathcal{C}_G \subseteq 2^E$ なので一般的に $|\mathcal{C}_G|$ は指数サイズ

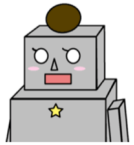
- $R_p(G) \coloneqq \sum_{X \in \mathcal{C}_G} p(X)$ ：$O(|\Delta_G|)$ で計算可能
- $\mathcal{C}_G \coloneqq \{X \subseteq E \mid G[X] \text{ is connected}\}$ ：フロンティア法で構成可能



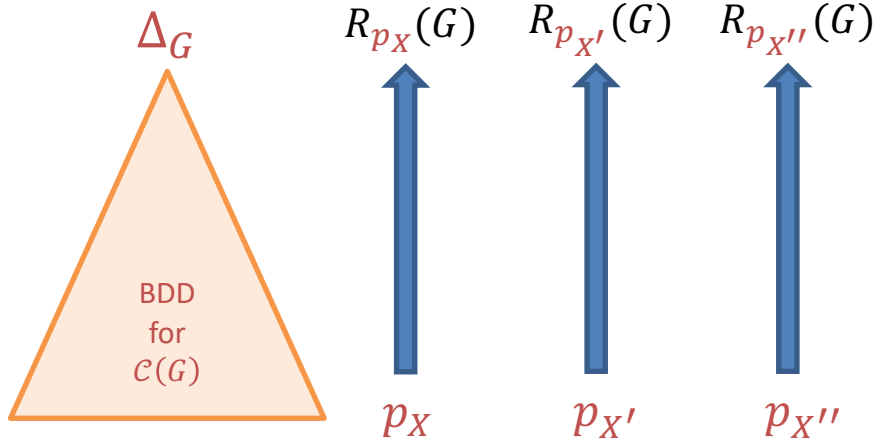


今回のDDポイント2：各$X \in \mathcal{C}_K$に対してBDDを構築するのは非現実的

- 目的
  - $X^* \coloneqq \mathrm{argmax}_{X \in \mathcal{C}_K} R_p(G[X])$
  - $\mathcal{C}_K \coloneqq \{X \subseteq E \mid |X| = K\}$

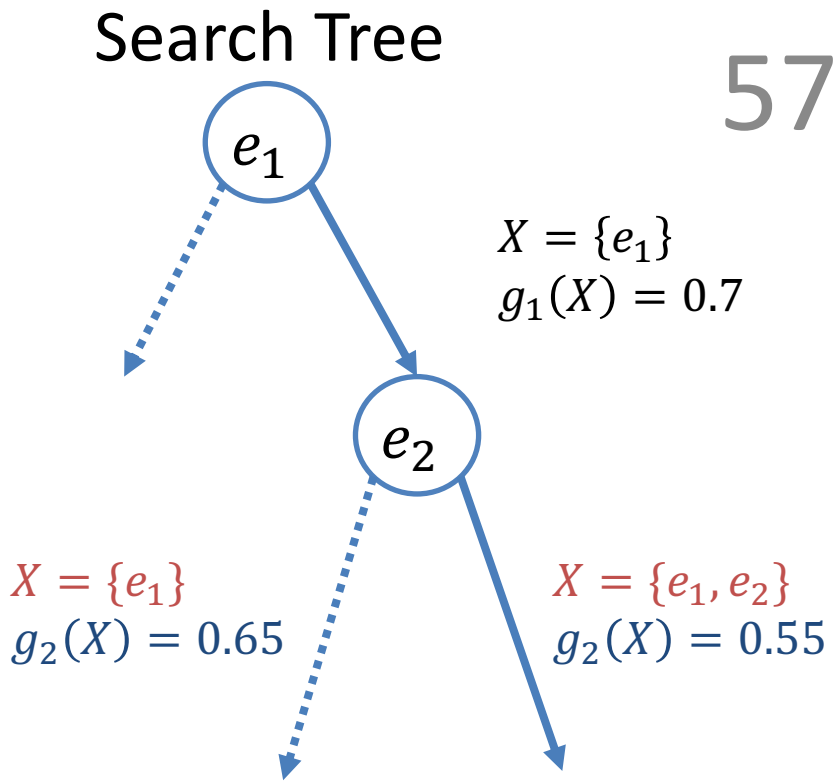


1. 候補グラフ $G[X]$ の信頼度 $R_p(G[X])$ を Δ_G 上で評価
   - $R_p(G[X]) = R_{p_X}(G)$
   - $p_X(e) \coloneqq \begin{cases} p(e) & e \in X \\ 0 & e \notin X \end{cases}$



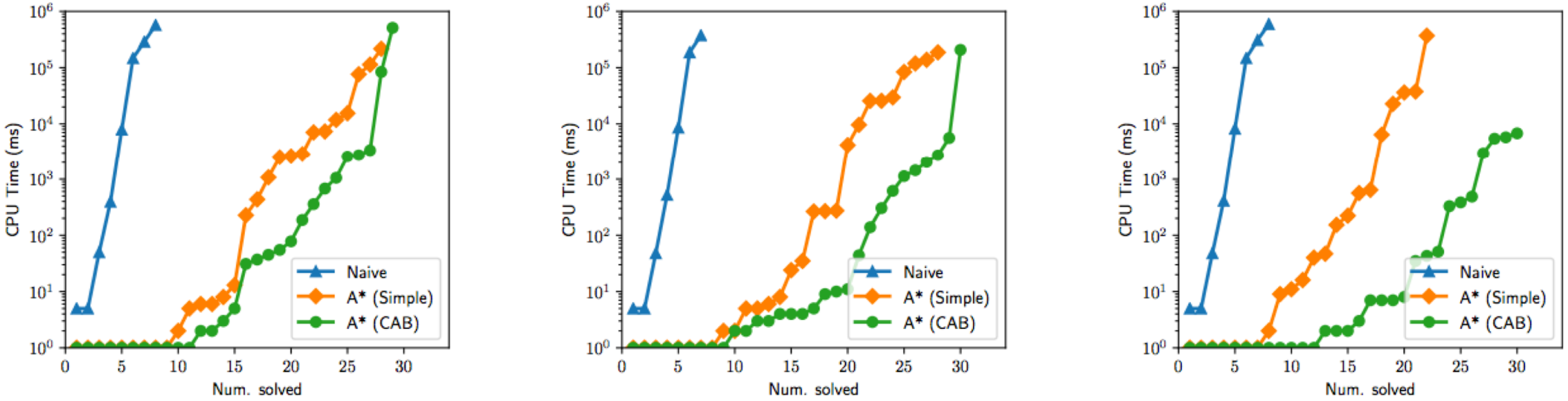
2. 現在の候補 X にあと L 本追加できるときの上限を計算（詳細略）

提案手法



1. Initialization:
    1. Construct Δ: the BDD for $\mathcal{C}_G$
    2. Add 1: $\langle 0, \emptyset \rangle$ to priority queue Q
2. Repeat: Enqueue $\langle i, X \rangle$ from Q
    1. Return X if $|X| = K$
    2. Evaluate $P \coloneqq g_{i+1}(X)$ on Δ and add P: $\langle i+1, X \rangle$ to Q
    3. Evaluate $P \coloneqq g_{i+1}(X \cup \{e_i\})$ on Δ and add P: $\langle i+1, X \cup \{e_i\}, \rangle$ to Q

実験



(a) $|T| = 2$ (b) $|T| = |V|/2$ (c) $|T| = |V|$

Fig. 2. Running time of exact methods for solving RMAX.

1枚概要

- ネットワーク信頼性
  - 通信ネットワークの各辺が確率的に故障するとき、通信が保たれる確率
  - ネットワーク信頼性の計算はNP困難
- 信頼性最大化問題
  - 限られた予算・設備内で、ネットワーク信頼性が最大の構成を見つける問題
  - ネットワーク信頼性が O(1) で計算できたとしてもNP困難
- BDD を用いた信頼性最大化問題に対する厳密アルゴリズムを提案
  - 最良優先探索により最適な構成を発見
  - ネットワーク信頼性と優先度の評価にBDDを利用

概要

離散構造処理技術を機械学習(っぽい)問題に適用した事例を紹介

1. **“A Dynamic Programming Algorithm for Tree Trimming-based Text Summarization”**

   Nishino, Yasuda, Hirao, Minato, Nagata, NAACL’2015.

2. **“Exact Computation of Influence Spread by Binary Decision Diagrams”**

   Maehara, Suzuki, Ishihata, WWW’2017.

3. **“Efficient Bandit Combinatorial Optimization Algorithm with Zero-suppressed Binary Decision Diagrams”**

   Sakaue, Ishihata, Minato, AISTATS’2018.

4. **“Optimizing Network Reliability via Best-First Search over Decision Diagrams”**（時間があれば…）

   Nishino, Inoue, Yasuda, Minato, Nagata, INFOCOM’2018.