離散最適化基礎論

第 14 回

前求解

岡本 吉央　(電気通信大学)

okamotoy@uec.ac.jp

2023 年 1 月 24 日

最終更新：2023 年 1 月 24 日　12:42

講義計画 (前半)

＜準備＞

1. 整数計画法と線形計画法 (10/4)
2. 線形計画法の復習 (1)：線形不等式系と凸多面集合 (10/11)

＊ 休み (体育祭) (10/18)

3. 線形計画法の復習 (2)：単体法と双対定理 (10/25)
4. 線形計画緩和 (11/1)

＜モデリング＞

5. 整数計画モデリング (1)：組合せ最適化問題 (11/8)
6. 整数計画モデリング (2)：より複雑な問題 (11/15)
7. 整数計画モデリング (3)：離接計画 (11/22)

講義計画 (後半)

<アルゴリズム>

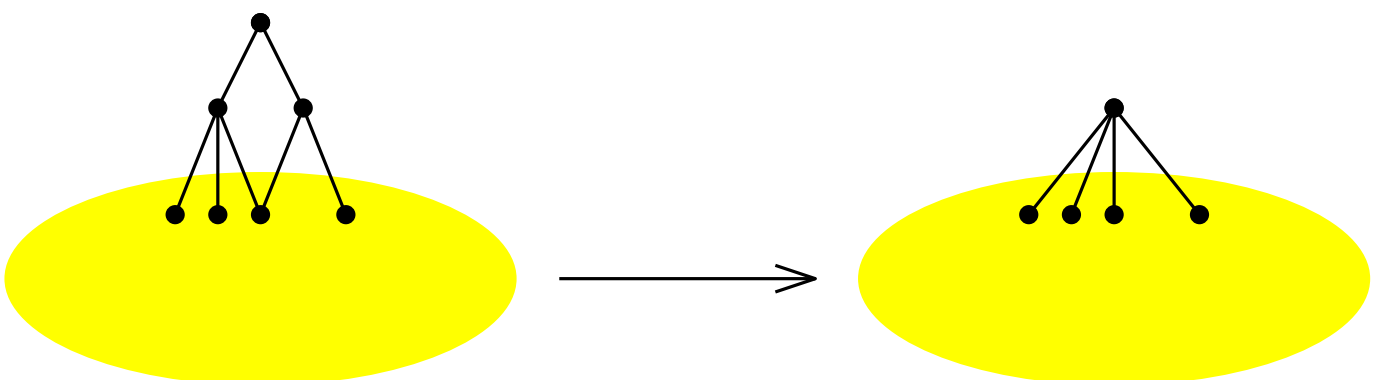
8. 分枝限定法 (11/29)
9. 切除平面法 (12/6)
10. 妥当不等式の追加 (12/13)
11. 列生成法 (12/20)
    * 休み (国内出張) (12/27)
    * 休み (冬季休業) (1/3)
12. ラグランジュ緩和 (1)：原理 (1/10)
13. ラグランジュ緩和 (2)：最適ラグランジュ緩和 (1/17)
14. 前求解 (1/24)

<まとめ>

15. 期末試験 (1/31)

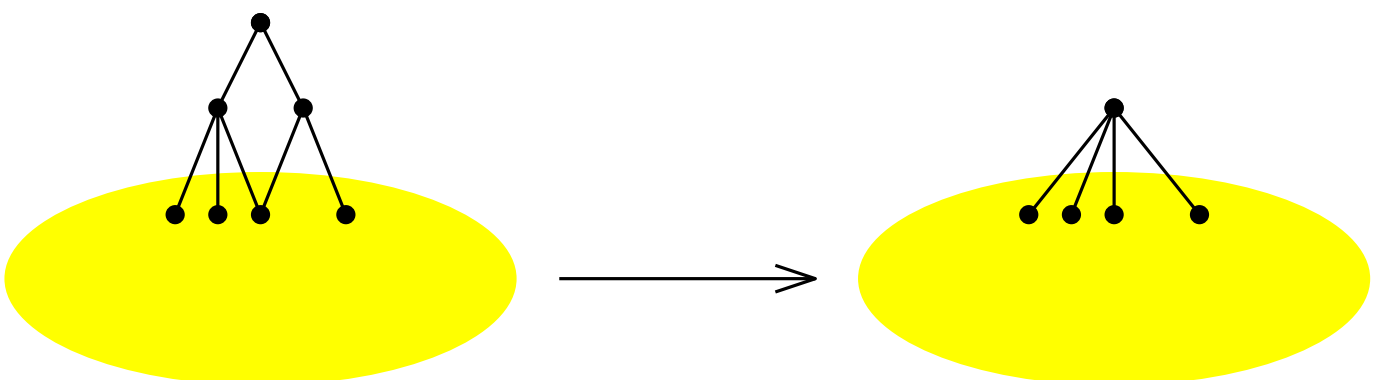
今日の内容

- 前求解とその威力
- 最大独立集合問題に対する前求解

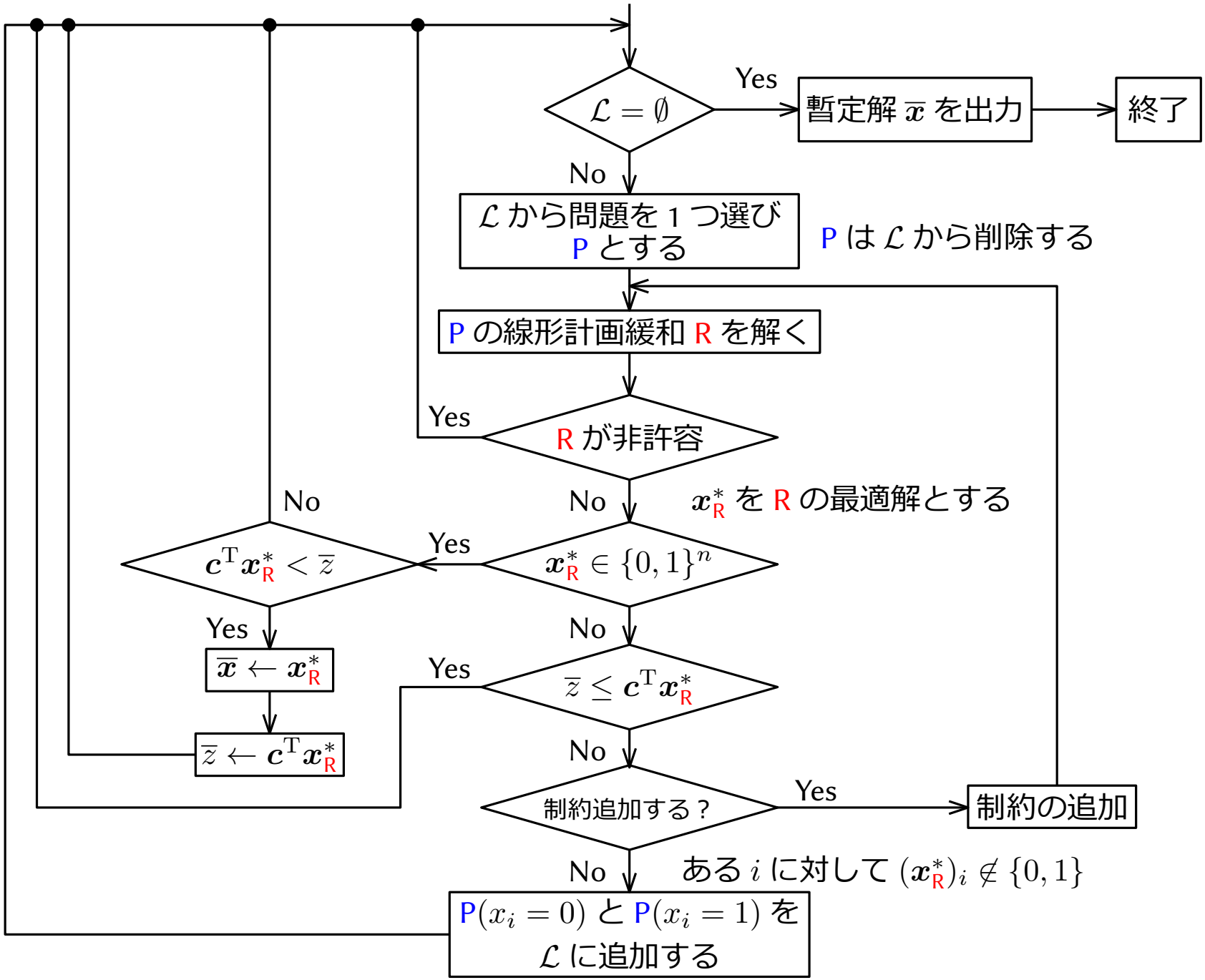


今日の内容

- 前求解とその威力
- 最大独立集合問題に対する前求解



(復習) 分枝切除法


$\mathcal{L} = \emptyset$
Yes
暫定解 $\overline{\boldsymbol{x}}$ を出力
終了
No
$\mathcal{L}$ から問題を 1 つ選び P とする
P は $\mathcal{L}$ から削除する
P の線形計画緩和 R を解く
Yes
R が非許容
No
$\boldsymbol{x}^*_{\mathrm{R}}$ を R の最適解とする
No
$\boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*_{\mathrm{R}} < \overline{z}$
Yes
$\boldsymbol{x}^*_{\mathrm{R}} \in \{0,1\}^n$
Yes
$\overline{\boldsymbol{x}} \leftarrow \boldsymbol{x}^*_{\mathrm{R}}$
No
$\overline{z} \leq \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*_{\mathrm{R}}$
Yes
$\overline{z} \leftarrow \boldsymbol{c}^{\mathrm{T}}\boldsymbol{x}^*_{\mathrm{R}}$
No
制約追加する？
Yes
制約の追加
No
ある i に対して $(\boldsymbol{x}^*_{\mathrm{R}})_i \notin \{0,1\}$
P$(x_i = 0)$ と P$(x_i = 1)$ を $\mathcal{L}$ に追加する

(復習) 分枝切除法：戦略 まとめ

(1) **分枝戦略** (branching strategy)
　　どの変数を使って分枝操作を行うか
(2) **節点選択戦略** (node selection strategy)
　　$\mathcal{L}$ の中からどの問題を選ぶか
(3) **カット生成戦略** (cut generation strategy)
　　どのような制約をどれだけ，いつ追加するか

戦略によって，分枝切除法のパフォーマンスは大きく変わる

入力の構造によって，よい戦略は変わりうる

(復習) 分枝切除法：戦略 まとめ

(1) **分枝戦略** (branching strategy)
　　どの変数を使って分枝操作を行うか
(2) **節点選択戦略** (node selection strategy)
　　$\mathcal{L}$ の中からどの問題を選ぶか
(3) **カット生成戦略** (cut generation strategy)
　　どのような制約をどれだけ，いつ追加するか

戦略によって，分枝切除法のパフォーマンスは大きく変わる

入力の構造によって，よい戦略は変わりうる

(4) **前求解戦略** (presolve strategy)
　　どのような前求解を行うか

時間があれば，最後の方の授業で扱うかもしれない

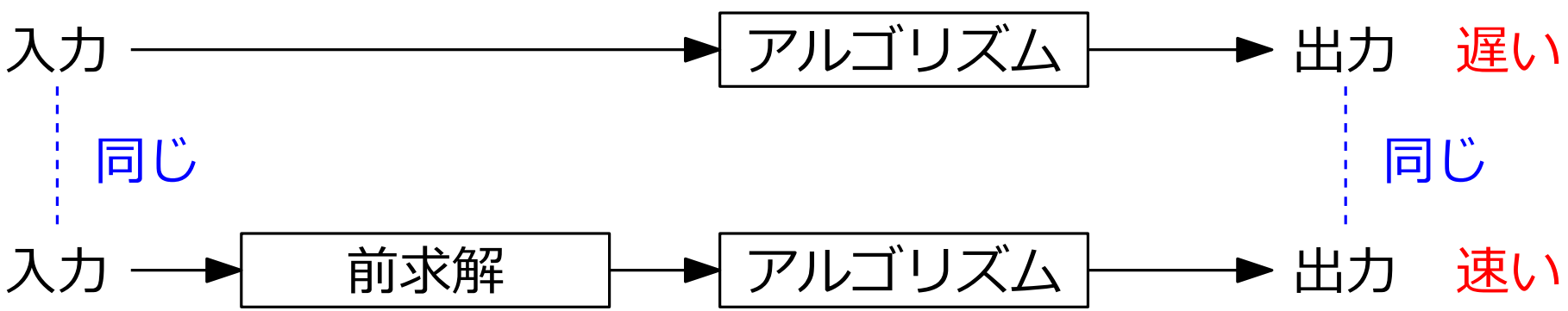
多くの最適化ソルバーでは，戦略の選択をパラメータで行える

前求解 (まえきゅうかい)

前求解 (presolve) とは？

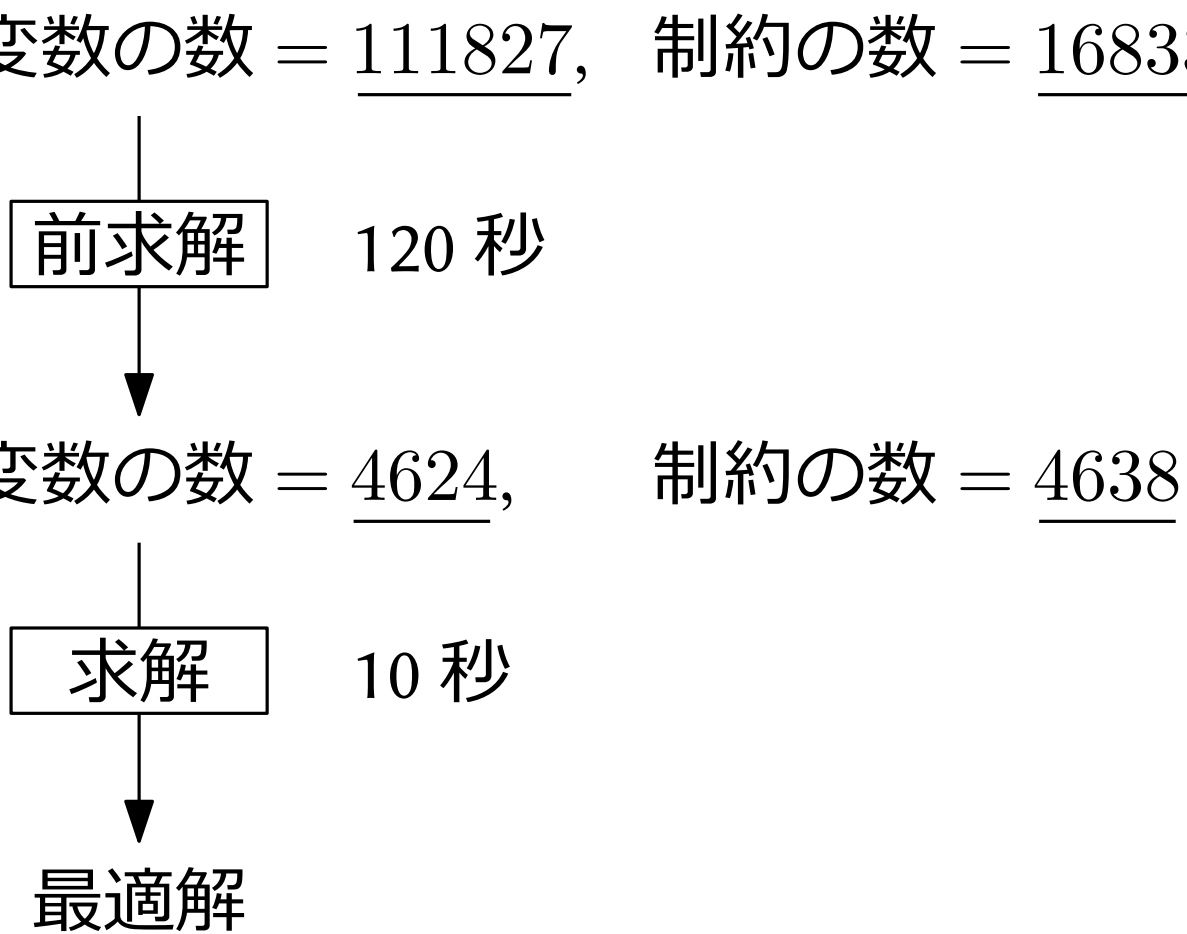
前求解 とは，入力を変換することによって，
等価な解きやすい入力を得ること

前処理 (preprocessing) と呼ばれることもあるが，
前処理という用語はデータの整形に対して用いられることも多い



前求解の威力 (1)

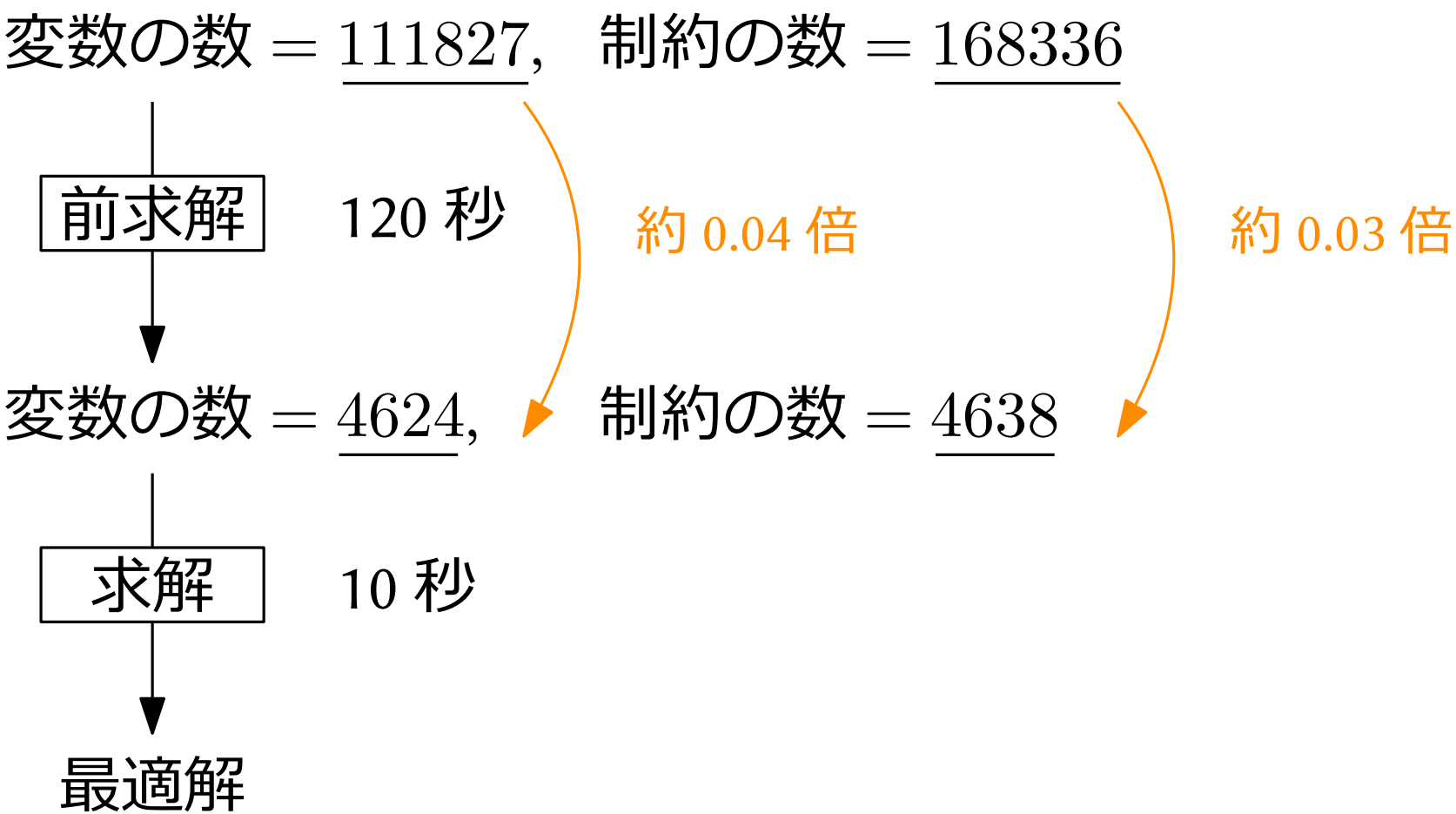
MIPLIB2017 のインスタンス physician… physician… physiciansched6-2.mps



ソルバー：SCIP 8.0.1, presolving emphasis aggressive
プロセッサ：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
メモリ：16.0 GB，OS：Windows 10 Home

前求解の威力 (1)

MIPLIB2017 のインスタンス physician… physician… physiciansched6-2.mps



ソルバー：SCIP 8.0.1, presolving emphasis aggressive
プロセッサ：Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz
メモリ：16.0 GB，OS：Windows 10 Home

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

ソルバー：CPLEX 12.5
プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ
メモリ：24 GB

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

前求解あり

前求解なし

ソルバー：CPLEX 12.5

プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ

メモリ：24 GB

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

時間制限：10k 秒

全インスタンス数

ソルバー：CPLEX 12.5

プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ

メモリ：24 GB

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

時間制限：10k 秒

3087 個中，解けなかったインスタンス数

ソルバー：CPLEX 12.5

プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ

メモリ：24 GB

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

時間制限：10k 秒

前求解なしの方が速く解けたインスタンス数

前求解なしの方が遅く解けたインスタンス数

ソルバー：CPLEX 12.5

プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ

メモリ：24 GB

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

時間制限：10k 秒

前求解なしの求解時間/前求解ありの求解時間

前求解なしのノード数/前求解ありのノード数

ソルバー：CPLEX 12.5

プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ

メモリ：24 GB

どちらも幾何平均

前求解の威力 (2)

Achterberg, Wunderling (2013) より抜粋

		default	no presolving				
bracket	models	tilim	tilim	faster	slower	time	nodes
all	3087	94	556	323	1841	3.50	3.97
[0, 10k]	3002	9	471	323	1841	3.63	4.20
[1, 10k]	2090	9	471	273	1703	6.27	7.18
[10, 10k]	1484	9	471	146	1295	11.40	13.19
[100, 10k]	1022	9	471	78	932	21.95	25.87
[1k, 10k]	687	9	471	33	648	43.28	49.00

どちらかが [1k, 10k] の間の時間 (秒) で解けたインスタンスに制限

ソルバー：CPLEX 12.5
プロセッサ：Intel Xeon CPU E5430 @ 2.66 GHz, 12 core のクラスタ
メモリ：24 GB

前求解のアルゴリズム

これまでに 前求解アルゴリズムが いろいろと提案されている

(1) 整数計画問題全般に使えるもの

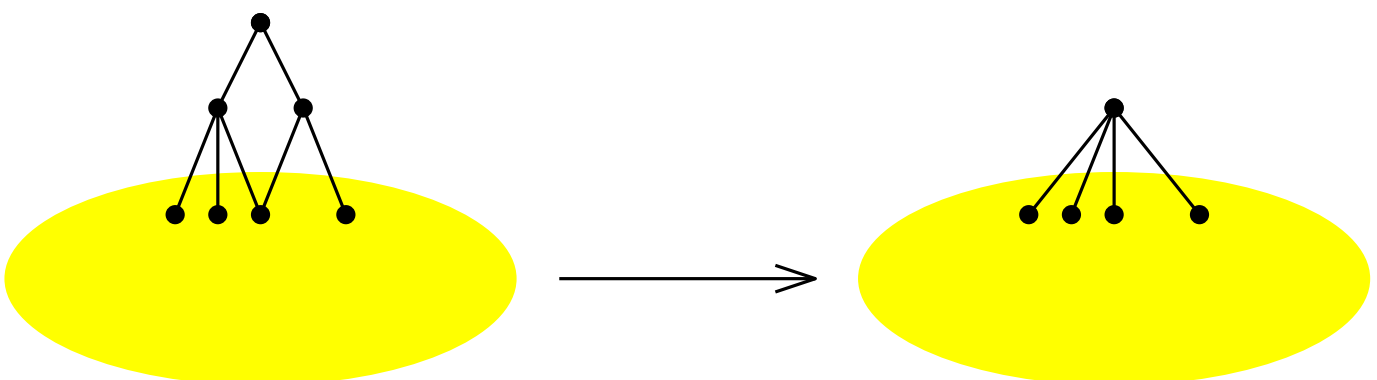
- Savelsbergh (1994)
- Achterberg (2007)
- Achterberg, Bixby, Gu, Rothberg, Weninger (2020)
- ...

(2) 特殊な問題に対して使えるもの

→ ここで **最大独立集合** について紹介する

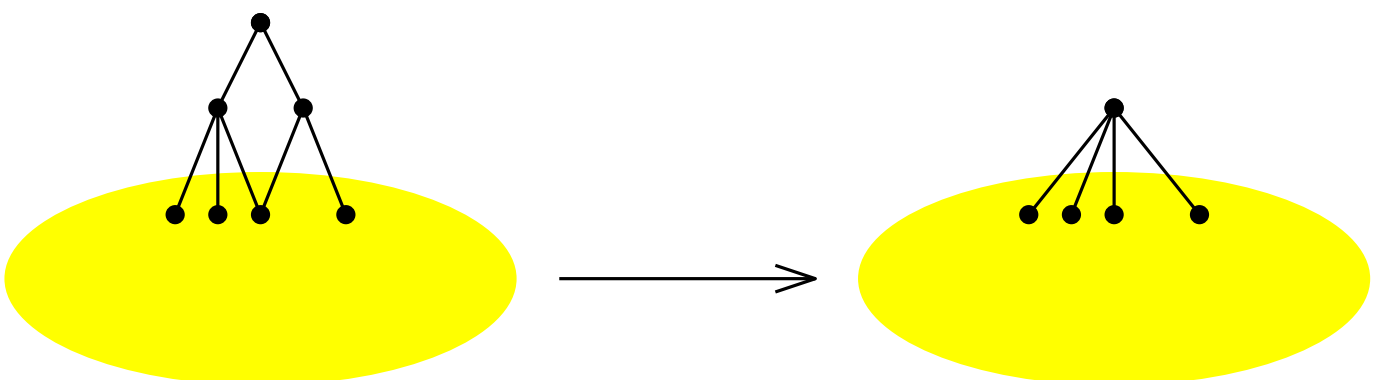
今日の内容

- 前求解とその威力
- 最大独立集合問題に対する前求解



今日の内容

- 前求解とその威力
- 最大独立集合問題に対する前求解



最大独立集合問題に対する前求解

いまから紹介する内容は次の論文に基づく

- D. Hespe, C. Schultz, D. Strash. Scalable Kernelization for Maximum Independent Sets. ACM Journal of Experimental Algorithmics 24(1), 2019, Art. 1.16.

この論文の内容

- 最大独立集合問題に対する**並列前求解アルゴリズム**の提案
- 実験による**評価** (グラフの頂点数，前求解時間等)

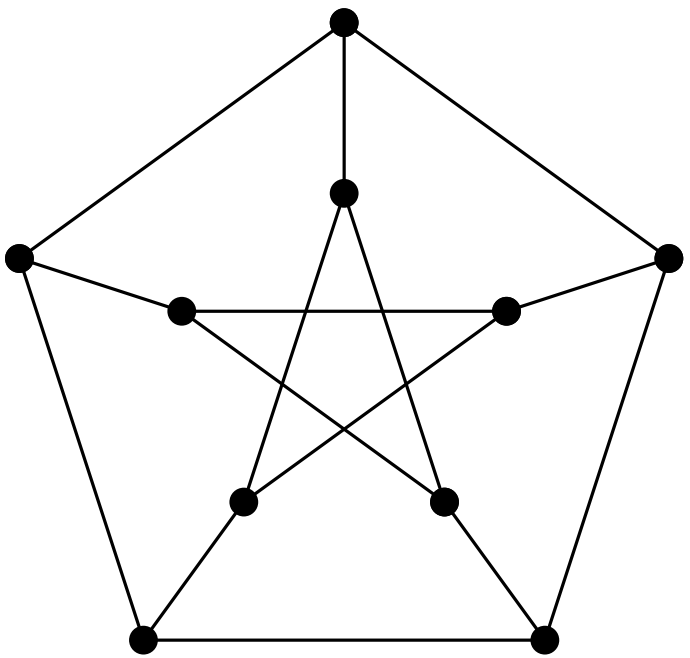
評価結果：抜粋

グラフ名	頂点数	先行研究 時間	先行研究 頂点数	本研究 時間	本研究 頂点数
uk-2002	19M	336.9	0.2M	11.8	0.3M
arabic-2005	23M	1,033.2	0.6M	25.7	0.6M
gsh-2015-tpd	31M	372.3	0.4M	32.0	0.5M
uk-2005	39M	131.9	0.9M	53.3	0.9M
it-2004	41M	499.7	1.7M	151.8	1.7M
sk-2005	51M	10,010.5	3.2M	178.3	3.5M
uk-2007-05	106M	18,829.4	3.5M	372.4	3.7M
webbase-2001	118M	4,207.8	0.7M	54.9	0.9M
asia.osm	12M	204.7	15.2K	1.2	34.9K
road_usa	24M	310.0	0.2M	4.1	0.2M
europe.osm	51M	302.4	8.4K	4.9	14.2K
rgg26	67M	9,887.7	49.6M	150.3	49.8M
rhg	100M	124.0	0	64.6	16
del24	17M	4,789.5	12.4M	515.	12.9M
del26	67M	20,728.7	49.9M	179.0	51.7M

先行研究：Akiba, Iwata (2016)

復習：最大独立集合問題

最大独立集合問題 (maximum independent set problem)



無向グラフ

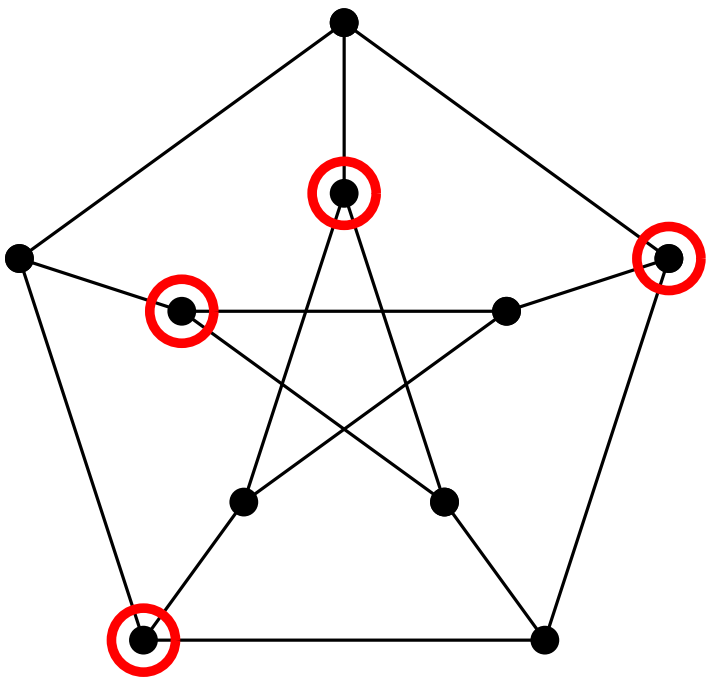
互いに隣接しない頂点を

できるだけ多く選びたい

独立集合 = 互いに隣接しない頂点からなる集合

復習：最大独立集合問題

最大独立集合問題 (maximum independent set problem)



無向グラフ

互いに隣接しない頂点を

できるだけ多く選びたい

独立集合 = 互いに隣接しない頂点からなる集合

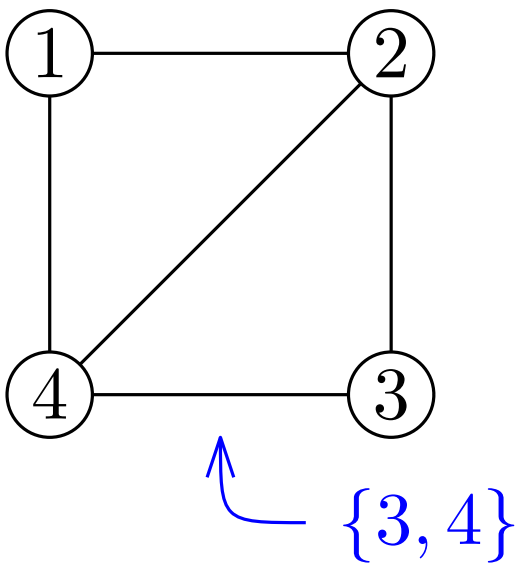
復習：無向グラフの定義と用語

無向グラフ $G = (V, E)$ (undirected graph)

頂点集合 (vertex set)　　辺集合 (edge set)

V の要素 = 頂点

E の要素 = 辺



$V = \{1, 2, 3, 4\}$

$E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

辺は $\{u, v\}$ を省略して uv と書くことも

復習：無向グラフの定義と用語

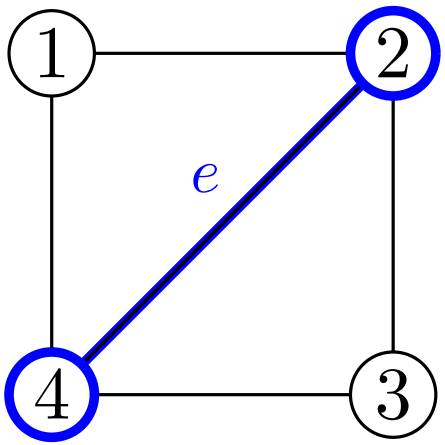
無向グラフ $G = (V, E)$ (undirected graph)

頂点集合 (vertex set)

辺集合 (edge set)

V の要素 = 頂点

E の要素 = 辺



$V = \{1, 2, 3, 4\}$

$E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

辺は $\{u, v\}$ を省略して uv と書くことも

辺 $e = \{u, v\}$ を考えると

- 2 頂点 u, v は **隣接** する
- 辺 e は 2 頂点 u, v に **接続** する
- 2 頂点 u, v は辺 e の **端点** である

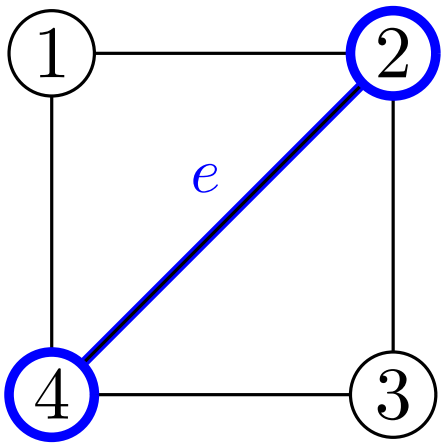
復習：無向グラフの定義と用語

無向グラフ $G = (V, E)$ (undirected graph)

頂点集合 (vertex set)　　辺集合 (edge set)

V の要素 = 頂点

E の要素 = 辺



$V = \{1, 2, 3, 4\}$

$E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$

辺は $\{u, v\}$ を省略して uv と書くことも

辺 $e = \{u, v\}$ を考えると

2 頂点 u, v は **隣接** する

辺 e は 2 頂点 u, v に **接続** する

2 頂点 u, v は辺 e の **端点** である

頂点 v の**次数** $\deg(v) = v$ に接続する辺の総数

今から行うこと

目標

最大独立集合問題に対する前求解アルゴリズムの紹介

- 頂点の折り畳み (vertex folding) (Chen et al. '01)
- 単体的頂点の除去 (Butenko et al. '07)
- 双子頂点の縮小 (Xiao, Nagamochi '13)

Hespe et al. の論文では，次のアルゴリズムも使われている

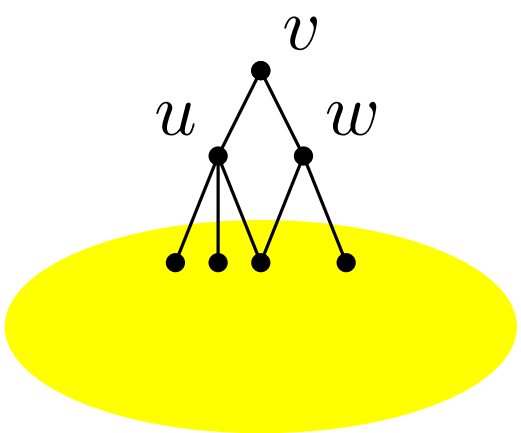
- 非拘束頂点 (unconfined vertex) の除去 (Xiao, Nagamochi '13)
- ダイアモンドの縮小 (Akiba, Iwata '16)
- LP 緩和の利用 (Nemhauser, Trotter '75)
- LinearTime, NearLinear (Chang et al. '17)

頂点の折り畳み：性質

無向グラフ $G = (V, E)$

頂点の折り畳み：考える状況

- 頂点 v の次数が 2　　(v は頂点 u, w と隣接)
- 頂点 u, w は隣接していない

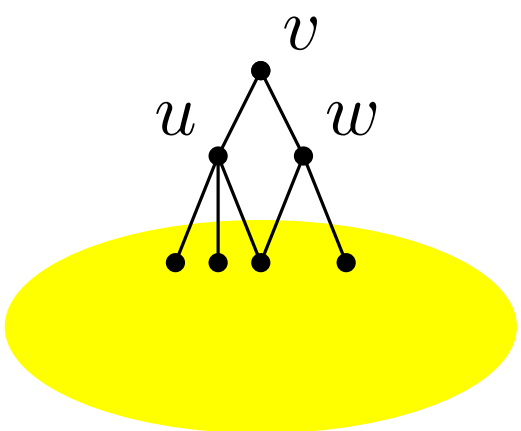


頂点の折り畳み：性質

無向グラフ $G = (V, E)$

頂点の折り畳み：考える状況

- 頂点 v の次数が 2　(v は頂点 u, w と隣接)
- 頂点 u, w は隣接していない



性質：次のどちらかを満たす最大独立集合 S が存在する

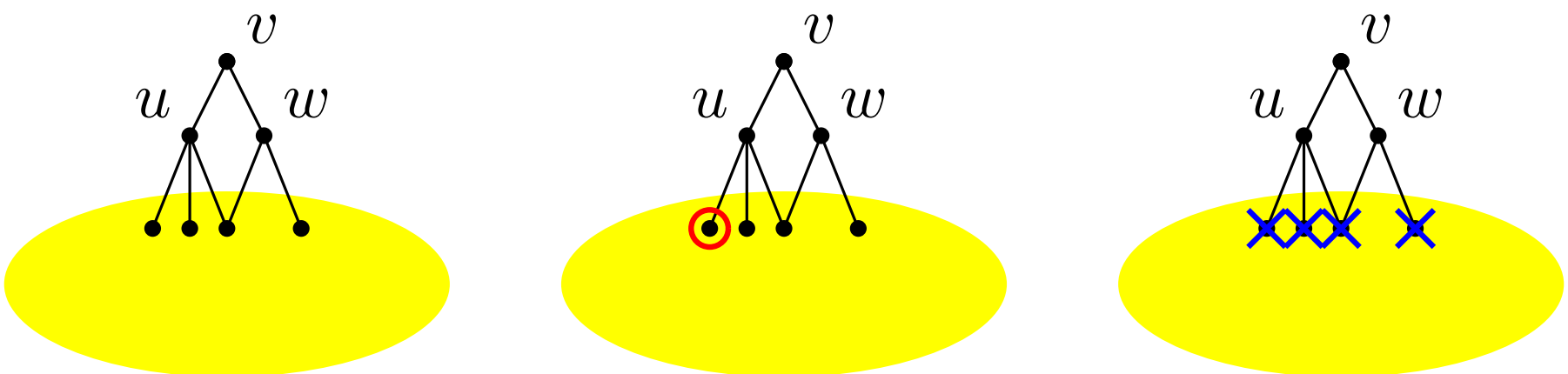
- $v \in S$
- $u \in S$ かつ $w \in S$

頂点の折り畳み：性質

無向グラフ $G = (V, E)$

頂点の折り畳み：考える状況

- 頂点 v の次数が 2　(v は頂点 u, w と隣接)
- 頂点 u, w は隣接していない



性質：次のどちらかを満たす最大独立集合 S が存在する

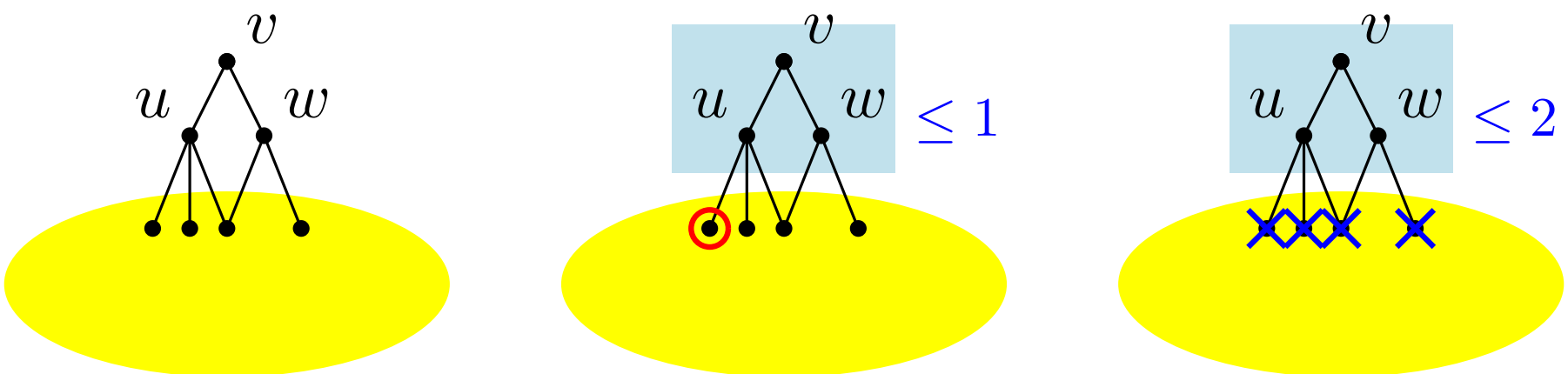
- $v \in S$
- $u \in S$ かつ $w \in S$

頂点の折り畳み：性質

無向グラフ $G = (V, E)$

頂点の折り畳み：考える状況

- 頂点 v の次数が 2　(v は頂点 u, w と隣接)
- 頂点 u, w は隣接していない



性質：次のどちらかを満たす最大独立集合 S が存在する

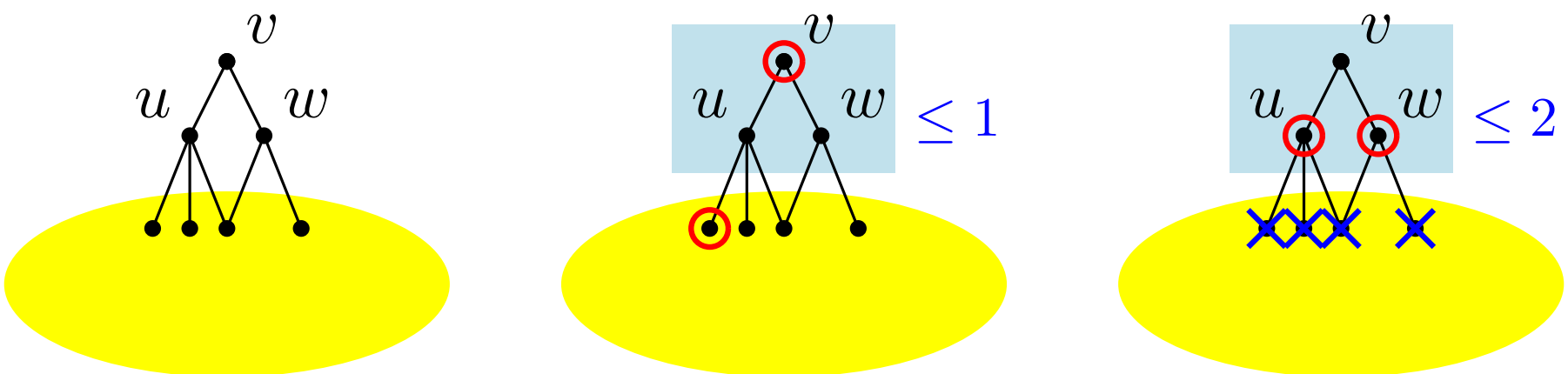
- $v \in S$
- $u \in S$ かつ $w \in S$

頂点の折り畳み：性質

無向グラフ $G = (V, E)$

頂点の折り畳み：考える状況

- 頂点 v の次数が 2　(v は頂点 u, w と隣接)
- 頂点 u, w は隣接していない



性質：次のどちらかを満たす最大独立集合 S が存在する

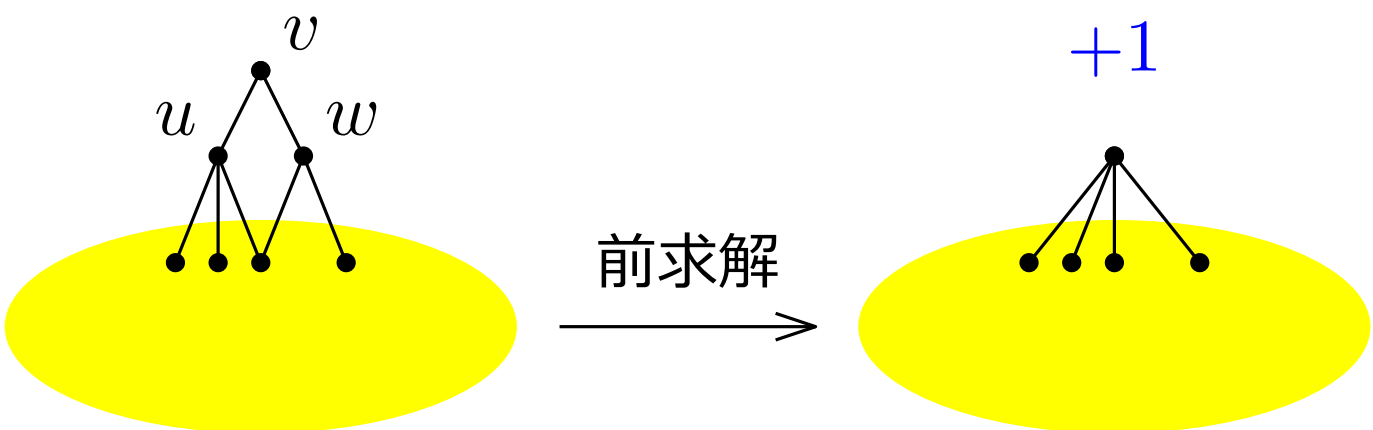
- $v \in S$
- $u \in S$ かつ $w \in S$

頂点の折り畳み：アルゴリズム

無向グラフ $G = (V, E)$

頂点の折り畳み：考える状況

- 頂点 v の次数が 2　(v は頂点 u, w と隣接)
- 頂点 u, w は隣接していない



頂点の折り畳み：アルゴリズム

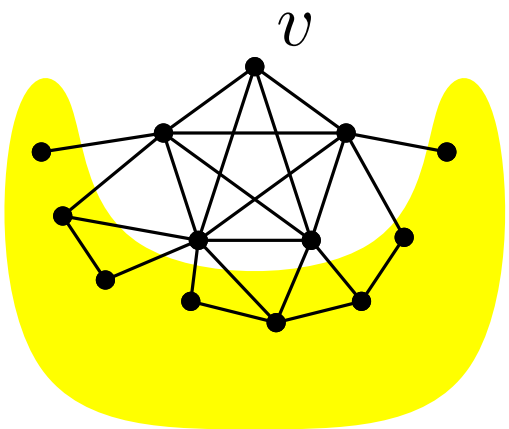
- v を除去して，u, w を同一視する

単体的頂点の除去：性質

無向グラフ $G = (V, E)$

定義：単体的頂点 (simplicial vertex)

頂点 v が **単体的** であるとは，
v に隣接する頂点同士が隣接していること



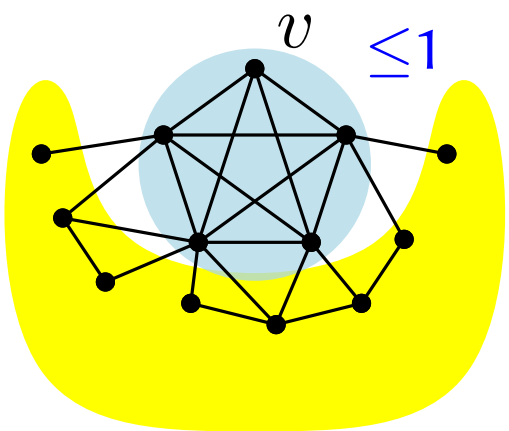
性質：v を要素として含む最大独立集合 S が存在する

単体的頂点の除去：性質

無向グラフ $G = (V, E)$

定義：単体的頂点 (simplicial vertex)

頂点 v が **単体的** であるとは，
v に隣接する頂点同士が隣接していること



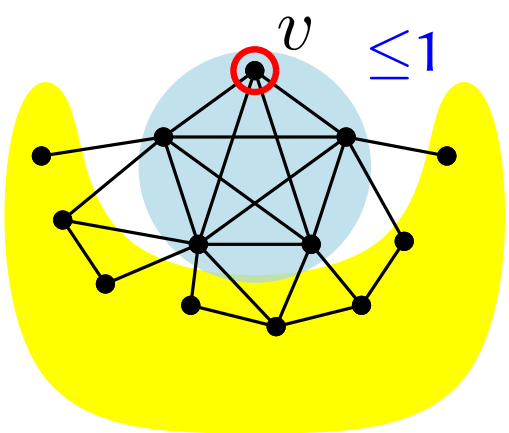
性質：v を要素として含む最大独立集合 S が存在する

単体的頂点の除去：性質

無向グラフ $G = (V, E)$

定義：単体的頂点 (simplicial vertex)

頂点 v が **単体的** であるとは，
v に隣接する頂点同士が隣接していること



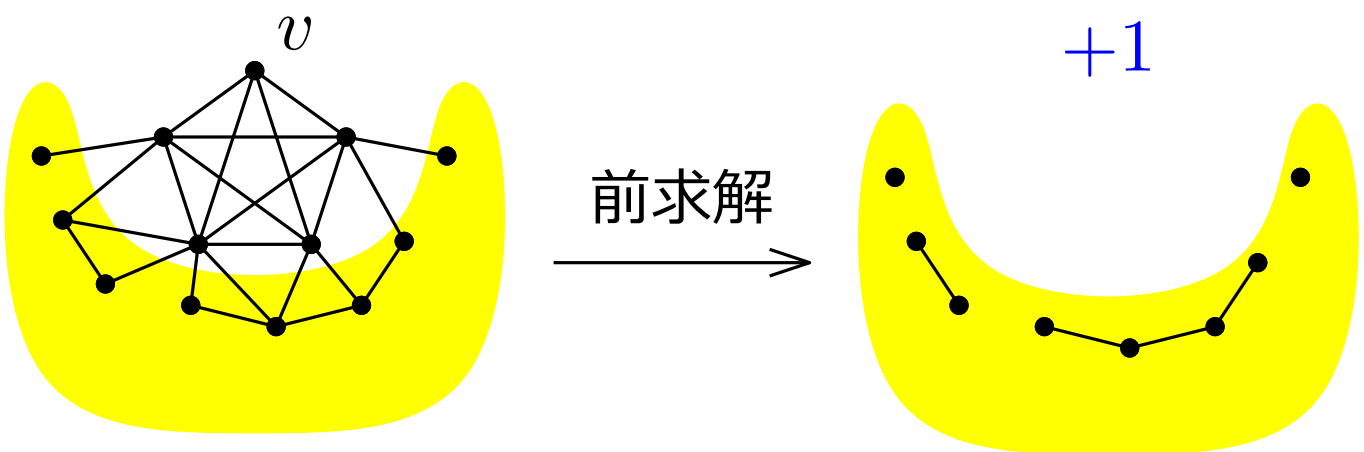
性質：v を要素として含む最大独立集合 S が存在する

単体的頂点の除去：アルゴリズム

無向グラフ $G = (V, E)$

定義：単体的頂点 (simplicial vertex)

頂点 v が **単体的** であるとは，
v に隣接する頂点同士が隣接していること



単体的頂点の除去：アルゴリズム

- 単体的頂点 v とその隣接頂点をすべて除去する

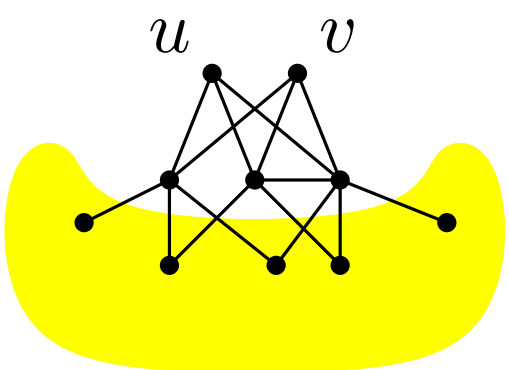
双子頂点の縮小：性質 (1)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



性質：u の隣接頂点間に辺がある $\Rightarrow$
u, v を要素として含む最大独立集合が存在する

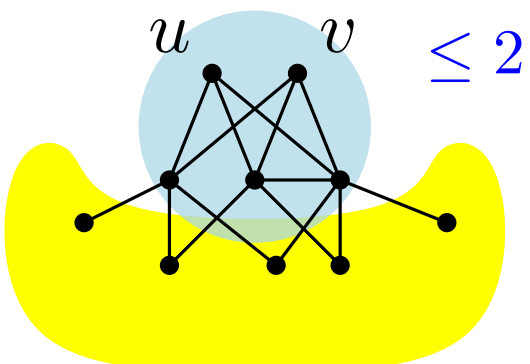
双子頂点の縮小：性質 (1)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



性質：u の隣接頂点間に辺がある $\Rightarrow$
u, v を要素として含む最大独立集合が存在する

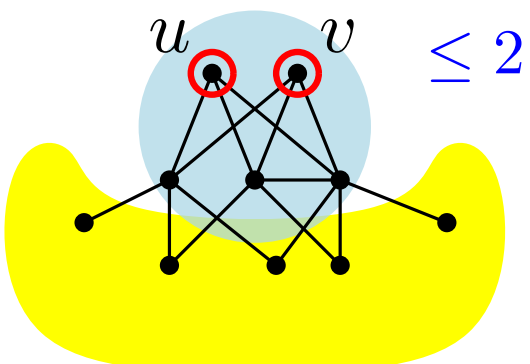
双子頂点の縮小：性質 (1)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



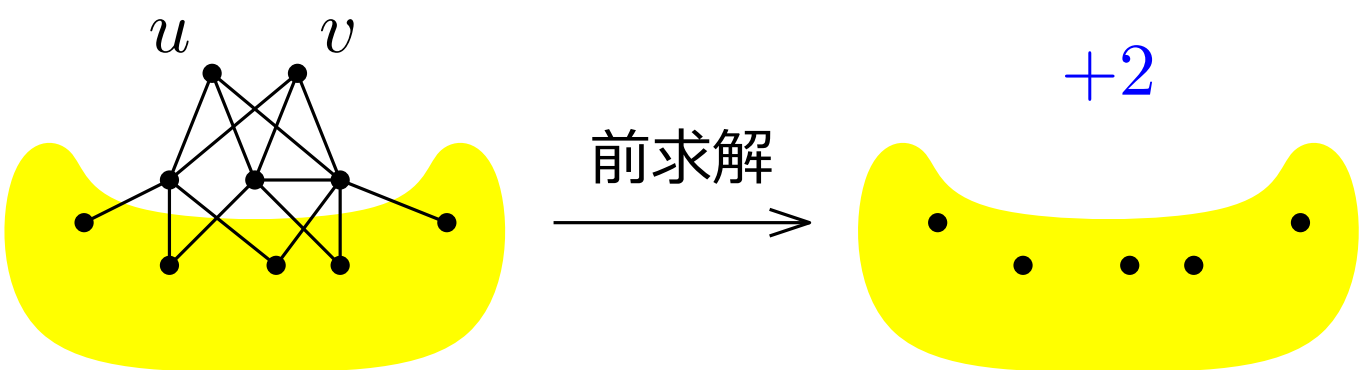
性質：u の隣接頂点間に辺がある $\Rightarrow$
u, v を要素として含む最大独立集合が存在する

双子頂点の縮小：アルゴリズム (1)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること



双子頂点の縮小：アルゴリズム (1)

u の隣接頂点間に辺があるとき
u, v と u の隣接頂点すべてを除去する

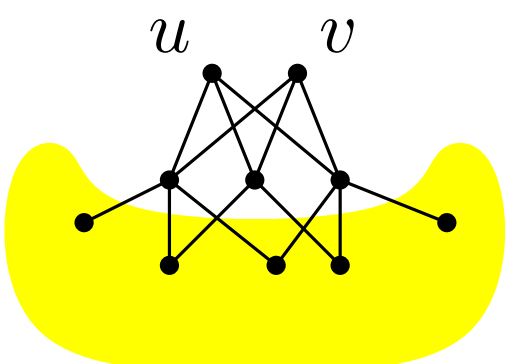
双子頂点の縮小：性質 (2)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



性質：u の隣接頂点間に辺がない $\Rightarrow$

次のどちらかを満たす最大独立集合 S が存在する

- $u, v \in S$
- u の隣接頂点すべて $\in S$

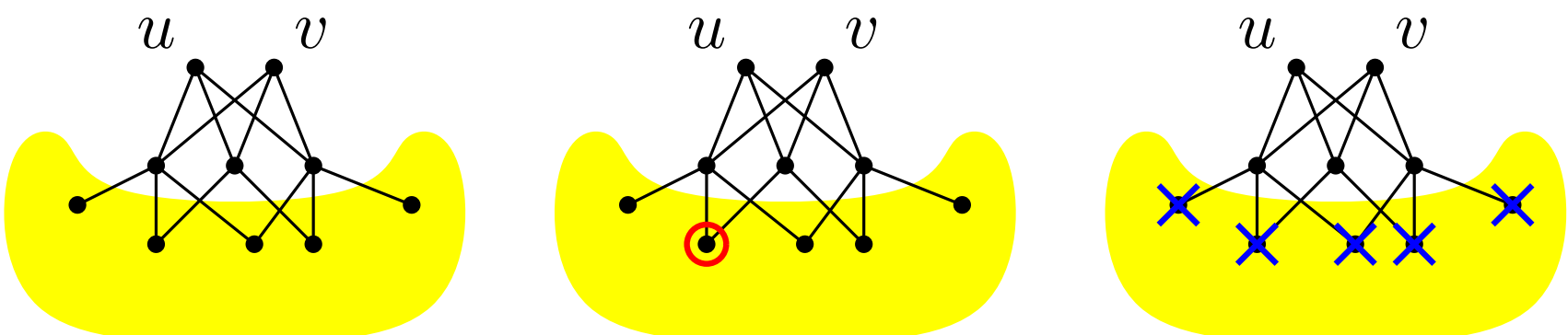
双子頂点の縮小：性質 (2)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



性質：u の隣接頂点間に辺がない $\Rightarrow$

次のどちらかを満たす最大独立集合 S が存在する

- $u, v \in S$
- u の隣接頂点すべて $\in S$

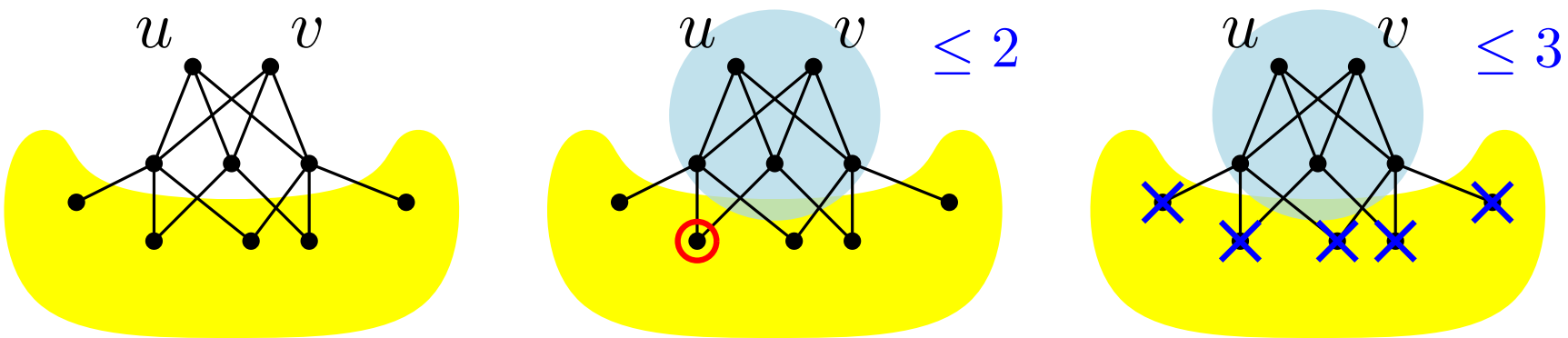
双子頂点の縮小：性質 (2)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



性質：u の隣接頂点間に辺がない $\Rightarrow$

次のどちらかを満たす最大独立集合 S が存在する

- $u, v \in S$
- u の隣接頂点すべて $\in S$

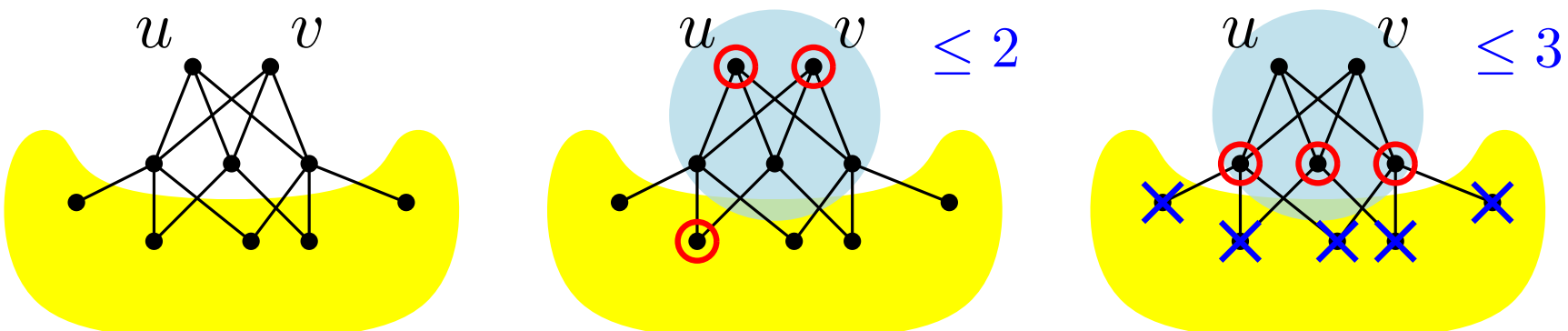
双子頂点の縮小：性質 (2)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



性質：u の隣接頂点間に辺がない $\Rightarrow$

次のどちらかを満たす最大独立集合 S が存在する

- $u, v \in S$
- u の隣接頂点すべて $\in S$

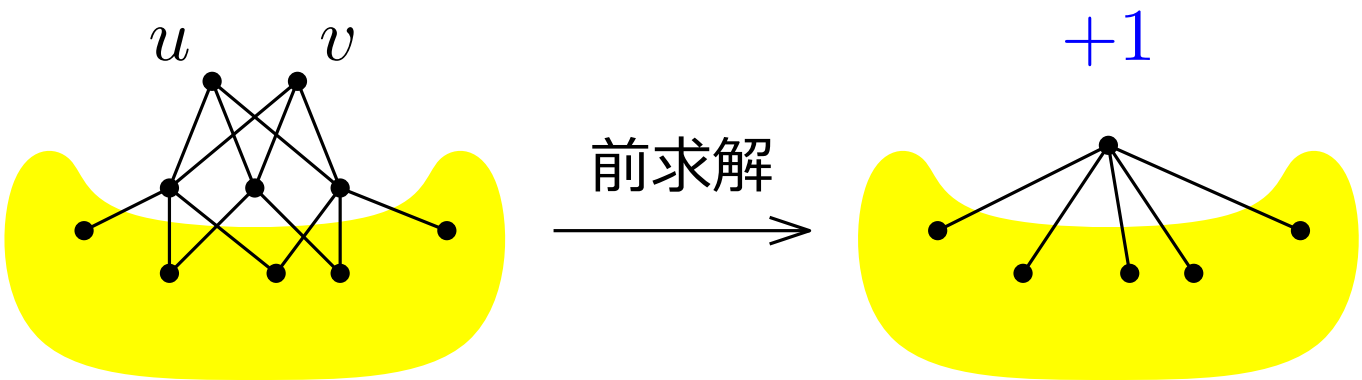
双子頂点の縮小：アルゴリズム (2)

無向グラフ $G = (V, E)$

定義：双子頂点 (twin vertices)

2 頂点 u, v が **双子** であるとは，$\deg(u) = \deg(v) = 3$ で，
u の隣接頂点集合 $= v$ の隣接頂点集合 であること

他の文献では，双子の定義が異なっているかもしれない



双子頂点の縮小：アルゴリズム (2)

u の隣接頂点間に辺がないとき，u, v と u の隣接頂点を除去し，
新たな頂点を追加し，u の隣接頂点の隣接頂点と隣接させる

最大独立集合問題に対する前求解

前求解アルゴリズム

次のアルゴリズムを適用できる限り適用する

- 頂点の折り畳み (Chen et al. '01)
- 単体的頂点の除去 (Butenko et al. '07)
- 双子頂点の縮小 (Xiao, Nagamochi '13)
- 非拘束頂点の除去 (Xiao, Nagamochi '13)
- ダイアモンドの縮小 (Akiba, Iwata '16)
- LP 緩和の利用 (Nemhauser, Trotter '75)
- LinearTime, NearLinear (Chang et al. '17)

Hespe et al. の論文では，単体的頂点の除去を行うのは
$\deg(v) \leq 2$ のときに限定している
非拘束頂点の除去も簡略化している

全体のまとめ

- この半年で学んだこと
- この半年で学ばなかったこと

講義概要

主題

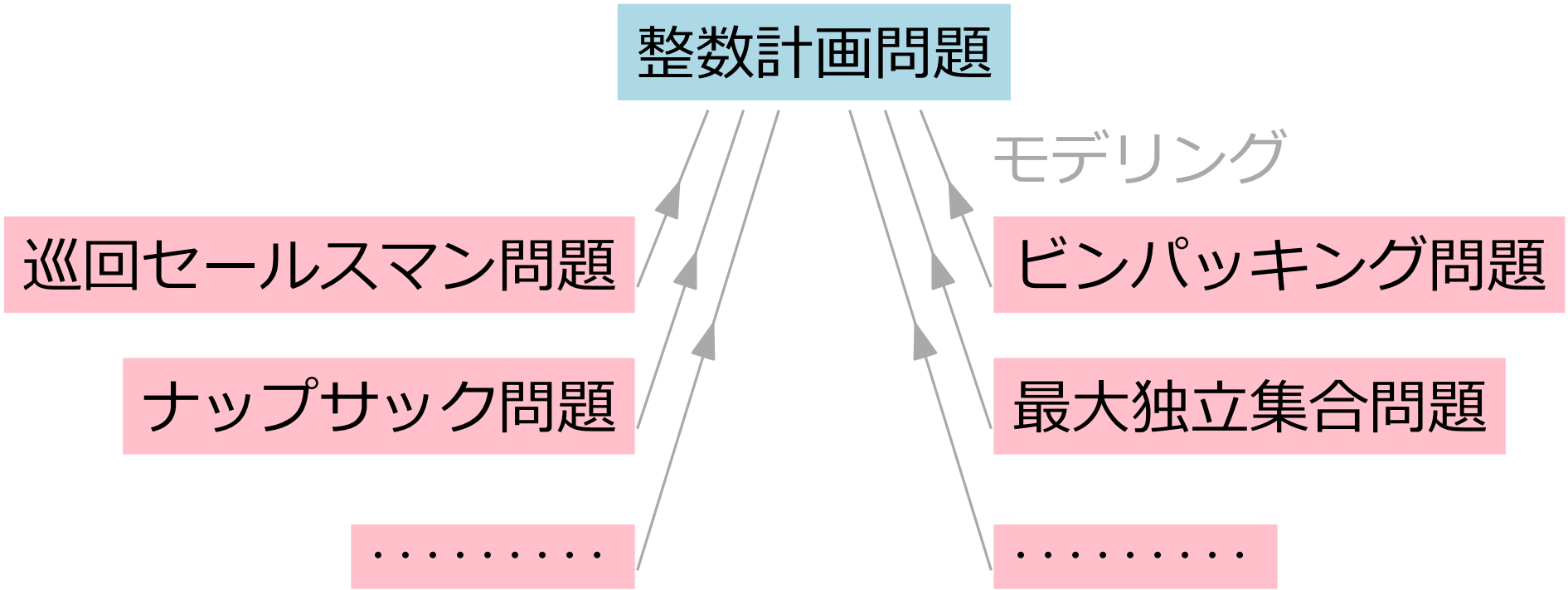
離散最適化における 1 つのトピックを集中的に学ぶ
今年度は「**整数計画法**」を題材とする

なぜ 整数計画法？

- 離散最適化における重要な手法である
- 実務上，多くの問題が整数計画法で解かれている

- おそらく，今後も多くの問題が整数計画法で解かれる
- そうでなくても，整数計画法で培われた技法は
  未来においても役立つ (はずである)

整数計画法：モデリング



考えるべき課題 (1)

整数計画問題として どのようにモデリングするか？

モデリング = モデル化 (modeling)，定式化 (formulation)

アルゴリズムをなぜ勉強するか？

考えるべき課題 (2)

整数計画問題を どのように解くか？

アルゴリズム

一般的 な枠組み　　個別問題 の性質

ソルバーをうまく使う　　自分で用意する

ポイント

大規模な問題を解くためには，どちらも重要

講義計画 (前半)

＜準備＞

1. 整数計画法と線形計画法 (10/4)
2. 線形計画法の復習 (1)：線形不等式系と凸多面集合 (10/11)

＊ 休み (体育祭) (10/18)

3. 線形計画法の復習 (2)：単体法と双対定理 (10/25)
4. 線形計画緩和 (11/1)

＜モデリング＞

5. 整数計画モデリング (1)：組合せ最適化問題 (11/8)
6. 整数計画モデリング (2)：より複雑な問題 (11/15)
7. 整数計画モデリング (3)：離接計画 (11/22)

講義計画 (後半)

<アルゴリズム>

8. 分枝限定法 (11/29)
9. 切除平面法 (12/6)
10. 妥当不等式の追加 (12/13)
11. 列生成法 (12/20)
    * 休み (国内出張) (12/27)
    * 休み (冬季休業) (1/3)
12. ラグランジュ緩和 (1)：原理 (1/10)
13. ラグランジュ緩和 (2)：最適ラグランジュ緩和 (1/17)
14. 前求解 (1/24)

<まとめ>

15. 期末試験 (1/31)

この半年で扱わなかったこと

整数計画法によるモデリング

- 凸多面体の整数性 (完全双対整数性など)
- …

整数計画法に対するアルゴリズム

- 他の切除平面 (lift-project 法など)
- 分解法 (Benders 分解，Dantzig-Wolfe 分解など)
- 主アルゴリズム (格子点の幾何学，Graver 基底など)
- 主ヒューリスティクス (feasibility pump など)
- 対称性の活用 (orbital fixing など)
- 並列化
- …

整数計画法のモデリング／アルゴリズムは現在も活発に研究されている