Algorytmy dokładne dla problemów NP-trudnych

Łukasz Kowalik

Instytut Informatyki, Uniwersytet Warszawski

Warszawa, 12-13.03.2010

Motywacja

Sposoby „radzenia sobie” z problemami NP-trudnymi

- Aproksymacja (dla problemów optymalizacji),
- FPT (parametryzacja),
- szczególne klasy egzemplarzy problemu,
- heurystyki

Motywacja

Sposoby „radzenia sobie” z problemami NP-trudnymi

- Aproksymacja (dla problemów optymalizacji),
- FPT (parametryzacja),
- szczególne klasy egzemplarzy problemu,
- ~~heurystyki~~

Motywacja

Sposoby „radzenia sobie” z problemami NP-trudnymi

- Aproksymacja (dla problemów optymalizacji),
- FPT (parametryzacja),
- szczególne klasy egzemplarzy problemu,
- ~~heurystyki~~

Niestety, te metody mają swoje ograniczenia

- Dla wiele ważnych problemów nie ma dobrych wielomianowych algorytmów aproksymacyjnych, o ile $\mathbf{P} \neq \mathbf{NP}$ (np: TSP, kolorowanie, klika)
- Wiele problemów parametryzowanych jest NP-trudnych (kolorowanie) lub W[1]-trudnych (klika)
- Nie zawsze egzemplarz należy do szczególnej klasy

Motywacja

Sposoby „radzenia sobie” z problemami NP-trudnymi

- Aproksymacja (dla problemów optymalizacji),
- FPT (parametryzacja),
- szczególne klasy egzemplarzy problemu,
- ~~heurystyki~~

A nawet jeśli działają, to oferują pewien kompromis:

Są szybkie, ale...

- niedokładne,
- tylko gdy parametr mały,
- tylko gdy egzemplarz specjalny,
- ~~niewiele wiadomo na pewno o tym, co algorytm zwraca~~

Ten wykład jest o...

Ten wykład jest o

algorytmice bez kompromisów ;)

tzn. dany problem NP-trudny chcemy po prostu rozwiązać i dążymy do jak najlepszego asymptotycznego czasu **pesymistycznego** (dla **dowolnych** egzemplarzy).

Mój pierwszy algorytm wykładniczy

Definicja

Niech G będzie grafem nieskierowanym. Zbiór $I \subseteq V$ jest niezależny, gdy żadne dwa wierzchołki I nie są połączone krawędzią w G.

Problem

Egzemplarz:
Graf nieskierowany $G = (V, E)$. Oznaczenie: $n = |V|$, $m = |E|$.

Problem:
Znaleźć zbiór niezależny I w G, t.ż. jeśli $J \subseteq V$ jest niezależny, to $|I| \geq |J|$.

Algorytm

Dla każdego z 2^n podzbiorów V sprawdź w czasie $O(n+m)$ czy jest on niezależny; zapamiętaj największy znaleziony zbiór.
Czas: $O((n+m)2^n)$; **pamięć:** wielomianowa.

Motywacja - cd.

- Będziemy badać jak bardzo możemy poprawić algorytmy brutalne.

Motywacja - cd.

- Będziemy badać jak bardzo możemy poprawić algorytmy brutalne.
- W przypadku większości problemów nie liczymy na algorytm $2^{o(n)}$: np. gdyby taki algorytm istniał dla problemu najliczniejszego zbioru niezależnego, to $\mathbf{SNP} \subseteq \mathbf{SUBEXP}$ (zdanie znacznie słabsze niż $\mathbf{P} = \mathbf{NP}$, ale również powszechnie uważane za nieprawdziwe).

Motywacja - cd.

- Będziemy badać jak bardzo możemy poprawić algorytmy brutalne.
- W przypadku większości problemów nie liczymy na algorytm $2^{o(n)}$: np. gdyby taki algorytm istniał dla problemu najliczniejszego zbioru niezależnego, to **SNP** $\subseteq$ **SUBEXP** (zdanie znacznie słabsze niż **P** $=$ **NP**, ale również powszechnie uważane za nieprawdziwe).
- Jeśli zamiast algorytmu $O(2^n)$ użyjemy algorytmu $O(1.189^n) = O(2^{n/4})$, oznacza to (z grubsza) że dysponując tym samym sprzętem możemy rozwiązywać egzemplarze 4 **razy** większe. Zauważmy, że 16-krotne przyspieszenie sprzętu oznacza (z grubsza), że możemy rozwiązywać egzemplarze większe o 4 wierzchołki.

Motywacja - cd.

- Będziemy badać jak bardzo możemy poprawić algorytmy brutalne.
- W przypadku większości problemów nie liczymy na algorytm $2^{o(n)}$: np. gdyby taki algorytm istniał dla problemu najliczniejszego zbioru niezależnego, to **SNP** $\subseteq$ **SUBEXP** (zdanie znacznie słabsze niż **P** $=$ **NP**, ale również powszechnie uważane za nieprawdziwe).
- Jeśli zamiast algorytmu $O(2^n)$ użyjemy algorytmu $O(1.189^n) = O(2^{n/4})$, oznacza to (z grubsza) że dysponując tym samym sprzętem możemy rozwiązywać egzemplarze 4 **razy** większe. Zauważmy, że 16-krotne przyspieszenie sprzętu oznacza (z grubsza), że możemy rozwiązywać egzemplarze większe o 4 wierzchołki.
- **Cel:** podać algorytm o złożoności $O(c^n)$, dla c możliwie małego.

Absurdy złożoności asymptotycznej

Wiemy, że

- $(n+m)2^n = o(2.0001^n)$,

Absurdy złożoności asymptotycznej

Wiemy, że

- $(n+m)2^n = o(2.0001^n)$,
- $n^{100}2^n = o(2.0001^n)$,

Absurdy złożoności asymptotycznej

Wiemy, że

- $(n+m)2^n = o(2.0001^n)$,
- $n^{100}2^n = o(2.0001^n)$,
- $n^{\log n}2^n = o(2.0001^n)$.

Absurdy złożoności asymptotycznej

Wiemy, że

- $(n+m)2^n = o(2.0001^n)$,
- $n^{100}2^n = o(2.0001^n)$,
- $n^{\log n}2^n = o(2.0001^n)$.

Dlatego wprowadzimy następującą notację:

definicja

$f(n) = O^*(g(n))$, gdy $f(n) = p(n)g(n)$ dla pewnego wielomianu p.

np $(n+m)2^n = O^*(2^n)$, $n^{100}2^n = O^*(2^n)$.

Algorytmy brutalne

Przypomnienie

Problem L jest w **NP**, gdy istnieje relacja $R(x, y)$ taka, że

- $R \in \mathbf{P}$,
- istnieje wielomian m taki, że dla każdego $(x, y) \in R$ jest $|y| \leq m(|x|)$,
- $L = \{x \ : \$ istnieje y takie, że $(x, y) \in R\}$.

Algorytm

Rozważmy następujący algorytm decydujący, czy $x \in L$:

```
1: for each y ∈ {0,1}*, |y| ≤ m(x) do
2:     if (x, y) ∈ R then
3:         return TRUE
4: return FALSE
```

Czas: $O^*(2^{m(x)})$, pamięć: wielomianowa.

Algorytmy brutalne: przykłady

SAT

Problem: Dla danej formuły logicznej o n zmiennych znaleźć wartościowanie spełniające.
Algorytm: $O^*(2^n)$.

k-kolorowanie

Problem: Dla danego grafu G i liczby $k \in \mathbb{N}$, znaleźć k-kolorowanie wierzchołkowe G, o ile istnieje.
Algorytm: $O^*(2^{\log kn}) = O^*(k^n)$.

Cykl Hamiltona

Problem: Dla danego grafu G znaleźć cykl Hamiltona, o ile istnieje.
Algorytm: $O^*(2^{n \log n}) = O^*(n^n)$; $O^*(n!)$.

Technika: rozgałęzianie (branching)

Najliczniejszy zbiór niezależny, ponownie

Oznaczenia

Sąsiedztwo $N(v) = \{x \ : \ vx \in E\}$,
Domknięte sąsiedztwo $N[v] = N(v) \cup \{v\}$.

Algorytm przez rozgałęzianie, v1.0

procedure MIS(G)

1: **if** $|V(G)| \leq 1$ **then**
2: **return** $|V(G)|$
3: $v \leftarrow$ dowolny wierzchołek G.
4: **if** $\deg(v) = 0$ **then**
5: **return** $1 + \text{MIS}(G - v)$
6: **else**
7: **return** $\max\{\text{MIS}(G - v), 1 + \text{MIS}(G - N[v])\}$

Żeby znaleźć moc najliczniejszego zbioru niezależnego, uruchamiamy MIS(G). (Łatwo przerobić ten algorytm, aby zwracał odpowiedni zbiór.)

Analiza złożoności czasowej

Niech D będzie drzewem rekurencyjnych wywołań procedury MIS. D ma wierzchołki trzech rodzajów:

- liście,
- wierzchołki posiadające dokładnie 1 syna (redukujące),
- wierzchołki posiadające ≥ 2 synów (rozgałęziające),

Analiza złożoności czasowej

Niech D będzie drzewem rekurencyjnych wywołań procedury MIS. D ma wierzchołki trzech rodzajów:

- liście,
- wierzchołki posiadające dokładnie 1 syna (redukujące),
- wierzchołki posiadające ≥ 2 synów (rozgałęziające),

Obserwacja

Liczba wszystkich węzłów drzewa D jest ograniczona przez wielomian od liczby liści D.

Analiza złożoności czasowej

Niech D będzie drzewem rekurencyjnych wywołań procedury MIS. D ma wierzchołki trzech rodzajów:

- liście,
- wierzchołki posiadające dokładnie 1 syna (redukujące),
- wierzchołki posiadające ≥ 2 synów (rozgałęziające),

Obserwacja

Liczba wszystkich węzłów drzewa D jest ograniczona przez wielomian od liczby liści D.

Wniosek

Złożoność czasowa algorytmu MIS wynosi $O^*(\ell)$, gdzie ℓ jest liczbą liści D.

Analiza złożoności czasowej

Szacujemy liczbę liści $T(n)$:

$$T(n) \leq T(n-1) + T(n-2)$$

$$T(0) = T(1) = 1$$

a więc $T(n) = F_{n+1} = O(1.62^n)$.

Analiza złożoności czasowej

Szacujemy liczbę liści $T(n)$:

$$T(n) \leq T(n-1) + T(n-2)$$

$$T(0) = T(1) = 1$$

a więc $T(n) = F_{n+1} = O(1.62^n)$.

Wniosek

Złożoność czasowa algorytmu MIS v1.0 wynosi $O(1.62^n)$.
(Dlaczego nie $O^*(1.62^n)$?)

Najliczniejszy zbiór niezależny, ponownie

Oznaczenia

Sąsiedztwo $N(v) = \{x \ : \ vx \in E\}$,
Domknięte sąsiedztwo $N[v] = N(v) \cup \{v\}$.

Algorytm przez rozgałęzianie, v2.0

procedure MIS(G)

1: **if** $|V(G)| \leq 1$ **then**
2: **return** $|V(G)|$
3: $v \leftarrow$ dowolny wierzchołek G.
4: **if** $\deg(v) \leq 1$ **then**
5: **return** $1 + \text{MIS}(G - v)$
6: **else**
7: **return** $\max\{\text{MIS}(G - v), 1 + \text{MIS}(G - N[v])\}$

Poprawność: Jeśli $\deg(v) = 1$ to istnieje najliczniejszy zbiór niezależny, który zawiera v.

MIS v2.0: Analiza złożoności czasowej

$$T(n) \leq T(n-1) + T(n-3)$$

$$T(0) = T(1) = 1$$

Jak to rozwiązać???

Rozwiązywanie rekurencji

Powiedzmy, że dla $s, r_i \in \mathbb{R}$, $k = O(1)$ mamy rekurencję w rodzaju:

$$T(s) = \sum_{i=1}^{k} \alpha_i T(s - r_i); \qquad T(s) = 1 \text{ dla } s \leq \max_i r_i.$$

Pokażemy przez indukcję, że $T(s) \leq \lambda^s$, dla pewnego $\lambda > 0$.
Baza indukcji: dla $0 \leq s \leq \max_i r_i$ OK.
Krok indukcyjny. Z założenia indukcyjnego:

$$T(s) \leq \sum_{i=1}^{k} \alpha_i \lambda^{s-r_i}$$

A więc wystarczy pokazać, że $\sum_{i=1}^{k} \alpha_i \lambda^{s-r_i} \leq \lambda^s$ co jest równoważne

$$\underbrace{1 - \sum_{i=1}^{k} \alpha_i \lambda^{-r_i}}_{f(\lambda)} \geq 0$$

Rozwiązywanie rekurencji

$$\underbrace{1 - \sum_{i=1}^{k} \alpha_i \lambda^{-r_i}}_{f(\lambda)} \geq 0$$

- $\lim_{x \to 0^+} f(x) = -\infty$

Rozwiązywanie rekurencji

$$\underbrace{1 - \sum_{i=1}^{k} \alpha_i \lambda^{-r_i}}_{f(\lambda)} \geq 0$$

- $\lim_{x \to 0^+} f(x) = -\infty$
- $\lim_{x \to +\infty} f(x) = 1$

Rozwiązywanie rekurencji

$$\underbrace{1 - \sum_{i=1}^{k} \alpha_i \lambda^{-r_i}}_{f(\lambda)} \geq 0$$

- $\lim_{x \to 0^+} f(x) = -\infty$
- $\lim_{x \to +\infty} f(x) = 1$
- czyli f ma dodatnie miejsce zerowe.

Rozwiązywanie rekurencji

$$\underbrace{1 - \sum_{i=1}^{k} \alpha_i \lambda^{-r_i}}_{f(\lambda)} \geq 0$$

- $\lim_{x \to 0^+} f(x) = -\infty$
- $\lim_{x \to +\infty} f(x) = 1$
- czyli f ma dodatnie miejsce zerowe.
- czyli nierówność jest spełniona jeśli λ jest największym dodatnim miejscem zerowym f.

Rozwiązywanie rekurencji

Wniosek

Jeśli, dla $s, r_i \in \mathbb{R}$, $k = O(1)$, mamy rekurencję w rodzaju:

$$T(s) = \sum_{i=1}^{k} \alpha_i T(s - r_i); \qquad T(s) = 1 \text{ dla } s \leq \max_i r_i.$$

to

$$T(s) \leq \lambda^s,$$

gdzie λ jest największym dodatnim miejscem zerowym funkcji

$$f(x) = 1 - \sum_{i=1}^{k} \alpha_i x^{-r_i}$$

(Te miejsca zerowe łatwo obliczamy na komputerze (bisekcja/R/matlab itp)

MIS v2.0: Analiza złożoności czasowej

$$T(n) \leq T(n-1) + T(n-3)$$

$$T(0) = T(1) = 1$$

~~Jak to rozwiązać???~~

Wniosek

Algorytm MIS v2.0 działa w czasie $O(1.4656^n)$.

Najliczniejszy zbiór niezależny, ponownie

Algorytm przez rozgałęzianie, v3.0

procedure MIS(G)

1: **if** $\max_{v \in V(G)} \deg(v) \leq 2$ **then**
2: Znajdź rozwiązanie w czasie wielomianowym i zwróć je.
3: $v \leftarrow$ wierzchołek o maksymalnym stopniu w G.
4: **return** $\max\{\text{MIS}(G - v), 1 + \text{MIS}(G - N[v])\}$

$$T(n) \leq T(n-1) + T(n-4)$$

Wniosek

Algorytm MIS v2.0 działa w czasie $O(1.3803^n)$.

Problem

Egzemplarz:

Formuła logiczna φ w postaci k-CNF o n zmiennych, tzn.

$$\phi = C_1 \wedge C_2 \wedge \ldots C_m,$$

$$C_i = (l_{i,1} \vee l_{i,2} \vee \ldots \vee l_{i,k}),$$

$$l_{i,j} = x_k \text{ lub } l_{i,j} = \neg x_k \text{dla pewnego } k \in \{1, \ldots, n\}$$

Problem:

Czy φ spełnialna, tzn. czy istnieje wartościowanie $v : \{x_1, \ldots x_n\} \rightarrow \{0, 1\}$ takie że $\varphi|_v = 1$.

k-SAT: algorytm

Rozważamy uogólnione formuły, w których literały mogą być stałymi 0 i 1.

Algorytm [Monien, Speckenmeyer 1985]

procedure SAT(φ)

1: **while** φ zawiera klauzulę C, która zawiera literał 1 **do**
2: Usuń klauzulę C z φ
3: **while** φ zawiera klauzulę C, która zawiera literał 0 **do**
4: **if** C ma tylko 1 literał **return** FALSE **else** Usuń literał 0 z C
5: **if** φ jest pusta **then**
6: **return** TRUE
7: $C \leftarrow$ dowolna klauzula w φ; $C = \ell_1 \vee \ldots \vee \ell_p$.
8: **return**
$\mathrm{SAT}(\varphi|_{\ell_1=1}) \vee \mathrm{SAT}(\varphi|_{\ell_1=0,\ell_2=1}) \vee \ldots \vee \mathrm{SAT}(\varphi|_{\ell_1=0,\ell_2=0,\ldots,\ell_{p-1}=0,\ell_p=1})$

Zapis $\varphi|x_i = 1$ lub $\varphi|\neg x_i = 0$ oznacza, że w całej formule φ literały x_i zastępujemy 1, a $\neg x_i$ zastępujemy 0; podobnie dla $\varphi|x_i = 0$ lub $\varphi|\neg x_i = 1$ literały x_i zastępujemy 0, a $\neg x_i$ zastępujemy 1;

k-SAT: analiza

$$
\begin{aligned}
T(n) \leq \max\{ & T(n-1), \\
& T(n-1) + T(n-2), \\
& T(n-1) + T(n-2) + T(n-3), \\
& \vdots \\
& T(n-1) + \ldots + T(n-k)\}
\end{aligned}
$$

Uwaga: Żeby oszacować $T(n)$ wystarczy rozwiązać każdą rekurencję osobno i wybrać najszybciej rosnącą z otrzymanych funkcji.

Wniosek

Przedstawiony algorytm działa w czasie $O^*(\beta_k^n)$, gdzie β_k jest największym dodatnim pierwiastkiem funkcji $f(x) = 1 - \sum_{i=1}^{k} x^{-i}$.
np. $\beta_3 \approx 1.8393$, czyli umiemy rozwiązać 3-SAT w czasie $O(1.84^n)$.

k-SAT: trochę lepszy algorytm

- naturalny pomysł: wybieramy zawsze najkrótszą klauzulę C.

k-SAT: trochę lepszy algorytm

- naturalny pomysł: wybieramy zawsze najkrótszą klauzulę C.
- początkowo może się zdarzyć, że $|C| = k$, ale wtedy w każdym z k wywołań rekurencyjnych pojawią się krótsze klauzule.

k-SAT: trochę lepszy algorytm

- naturalny pomysł: wybieramy zawsze najkrótszą klauzulę C.
- początkowo może się zdarzyć, że $|C| = k$, ale wtedy w każdym z k wywołań rekurencyjnych pojawią się krótsze klauzule.
- jeśli (gdzieś w środku drzewa rekursji) okaże się, że wszystkie klauzule mają długość k, oznacza to, że aktualna formuła φ jest spełnialna wtedy i tylko wtedy gdy wejściowa formuła φ_0 jest spełnialna. Wtedy przerywamy rekurencję i uruchamiamy algorytm na mniejszej formule φ.

k-SAT: trochę lepszy algorytm

- naturalny pomysł: wybieramy zawsze najkrótszą klauzulę C.
- początkowo może się zdarzyć, że $|C| = k$, ale wtedy w każdym z k wywołań rekurencyjnych pojawią się krótsze klauzule.
- jeśli (gdzieś w środku drzewa rekursji) okaże się, że wszystkie klauzule mają długość k, oznacza to, że aktualna formuła φ jest spełnialna wtedy i tylko wtedy gdy wejściowa formuła φ_0 jest spełnialna. Wtedy przerywamy rekurencję i uruchamiamy algorytm na mniejszej formule φ.
- W ten sposób uruchomimy algorytm co najwyżej n razy, za każdym razem algorytm co najwyżej raz rozgałęzia się na formule rozmiaru k.

k-SAT: trochę lepszy algorytm

- naturalny pomysł: wybieramy zawsze najkrótszą klauzulę C.
- początkowo może się zdarzyć, że $|C| = k$, ale wtedy w każdym z k wywołań rekurencyjnych pojawią się krótsze klauzule.
- jeśli (gdzieś w środku drzewa rekursji) okaże się, że wszystkie klauzule mają długość k, oznacza to, że aktualna formuła φ jest spełnialna wtedy i tylko wtedy gdy wejściowa formuła φ_0 jest spełnialna. Wtedy przerywamy rekurencję i uruchamiamy algorytm na mniejszej formule φ.
- W ten sposób uruchomimy algorytm co najwyżej n razy, za każdym razem algorytm co najwyżej raz rozgałęzia się na formule rozmiaru k.

Wniosek [Monien, Speckenmeyer 1985]

Przedstawiony algorytm działa w czasie $O^*({\beta_{k-1}}^n)$, gdzie β_k jest największym dodatnim pierwiastkiem funkcji $f(x) = 1 - \sum_{i=1}^{k} x^{-i}$.
Czyli umiemy rozwiązać 3-SAT w czasie $O(1.62^n)$.

Uwaga notacyjna

Często wygodnie jest używać pojęcia **wektora rozgałęzienia**.

Jeśli algorytm wykonuje ℓ wywołań rekurencyjnych oraz przy i-tym wywołaniu rozmiar problemu redukuje się o co najmniej r_i, to tej sytuacji odpowiada **wektor rozgałęzienia** $(r_1, r_2, \ldots, r_\ell)$.

Np. w ostatnim algorytmie możliwe wektory rozgałęzienia to $(1, 2)$, $\ldots$, $(1, 2, \ldots, k-1)$.

Dla danego wektora $(r_1, r_2, \ldots, r_\ell)$ definiujemy jego **czynnik pracy** $\lambda(r_1, \ldots, r_\ell)$ jako największe dodatnie miejsce zerowe funkcji $f(x) = 1 - \sum_{i=1}^{l} x^{-r_i}$.

Wniosek

Algorytm przez rozgałęzienia ma złożoność $O^*(\lambda^n)$, gdzie λ jest największym czynnikiem pracy dla możliwich wektorów rozgałęzienia.

Zliczanie maksymalnych zbiorów niezależnych

Definicja

Zbiór $I \subseteq V$ jest maksymalnym zbiorem niezależnym w $G = (V, E)$, gdy nie istnieje zbiór niezależny $J \supsetneq I$.

Zliczanie maksymalnych zbiorów niezależnych

Definicja

Zbiór $I \subseteq V$ jest maksymalnym zbiorem niezależnym w $G = (V, E)$, gdy nie istnieje zbiór niezależny $J \supsetneq I$.

Problem

Egzemplarz:
Graf nieskierowany $G = (V, E)$. Oznaczenie: $n = |V|$, $m = |E|$.

Problem:
Znaleźć liczbę maksymalnych zbiorów niezależnych w G.

Zliczanie maksymalnych zbiorów niezależnych

Algorytm

procedure #MIS(G)

1: **if** $|V(G)| = 0$ **then**
2: **return** 1
3: $v \leftarrow$ wierzchołek minimalnego stopnia w G.
4: Niech $v_0 = v$ oraz $N(v) = \{v_0, \ldots, v_{\deg(v)}\}$.
5: **return** $\sum_{i=0}^{\deg(v)} \#\text{MIS}(G - \{v_0, \ldots, v_i\} - N[v_i])$

Poprawność:

- Albo v jest w maksymalnym zbiorze niezależnym,
- albo v nie jest w maksymalnym zbiorze niezależnym, ale v_1 jest,
- albo v, v_1 nie są w maksymalnym zbiorze niezależnym, ale v_2 jest,
- ...
- albo $v, v_1, \ldots, v_{deg(v)-1}$ nie są w maksymalnym zbiorze niezależnym, ale $v_{\deg(v)}$ jest,

MIS: analiza

- Jeśli $\deg(v) = 0$ to nie ma rozgałęziania.

MIS: analiza

- Jeśli $\deg(v) = 0$ to nie ma rozgałęziania.
- Jeśli $\deg(v) = d \geq 1$ to wektor rozgałęzienia $(\underbrace{d+1, \ldots, d+1}_{d+1 \text{ razy}})$.

MIS: analiza

- Jeśli $\deg(v) = 0$ to nie ma rozgałęziania.
- Jeśli $\deg(v) = d \geq 1$ to wektor rozgałęzienia $(\underbrace{d+1, \ldots, d+1}_{d+1 \text{ razy}})$.
- $\lambda(\underbrace{d, \ldots, d}_{d \text{ razy}}) = d^{1/d}$.

MIS: analiza

- Jeśli $\deg(v) = 0$ to nie ma rozgałęziania.
- Jeśli $\deg(v) = d \geq 1$ to wektor rozgałęzienia $(\underbrace{d+1, \ldots, d+1}_{d+1 \text{ razy}})$.
- $\lambda(\underbrace{d, \ldots, d}_{d \text{ razy}}) = d^{1/d}$.
- $\frac{d}{dx} x^{1/x} = \frac{1-\ln(x)}{x^2} x^{1/x}$,

MIS: analiza

- Jeśli $\deg(v) = 0$ to nie ma rozgałęziania.
- Jeśli $\deg(v) = d \geq 1$ to wektor rozgałęzienia $(\underbrace{d+1, \ldots, d+1}_{d+1 \text{ razy}})$.
- $\lambda(\underbrace{d, \ldots, d}_{d \text{ razy}}) = d^{1/d}$.
- $\frac{d}{dx} x^{1/x} = \frac{1-\ln(x)}{x^2} x^{1/x}$,
- czyli $x^{1/x}$ ma maksimum dla $x = e$, dla $x > e$ maleje.

MIS: analiza

- Jeśli $\deg(v) = 0$ to nie ma rozgałęziania.
- Jeśli $\deg(v) = d \geq 1$ to wektor rozgałęzienia $(\underbrace{d+1, \ldots, d+1}_{d+1 \text{ razy}})$.
- $\lambda(\underbrace{d, \ldots, d}_{d \text{ razy}}) = d^{1/d}$.
- $\frac{d}{dx} x^{1/x} = \frac{1-\ln(x)}{x^2} x^{1/x}$,
- czyli $x^{1/x}$ ma maksimum dla $x = e$, dla $x > e$ maleje.
- tymaczasem $2^{1/2} \approx 1.41$ oraz $3^{1/3} \approx 1.44$.

Wniosek [Moon, Moser 1965]

Graf o n wierzchołkach ma co najwyżej $3^{n/3}$ zbiorów niezależnych. Można je wszystkie wypisać w czasie $O^*(k)$, gdzie k jest liczbą tych zbiorów.

(3,2)-CSP (Problem spełnialności więzów)

Następujący problem jest wspólnym uogólnieniem problemu 3-kolorowania i problemu 3-SAT.

Problem

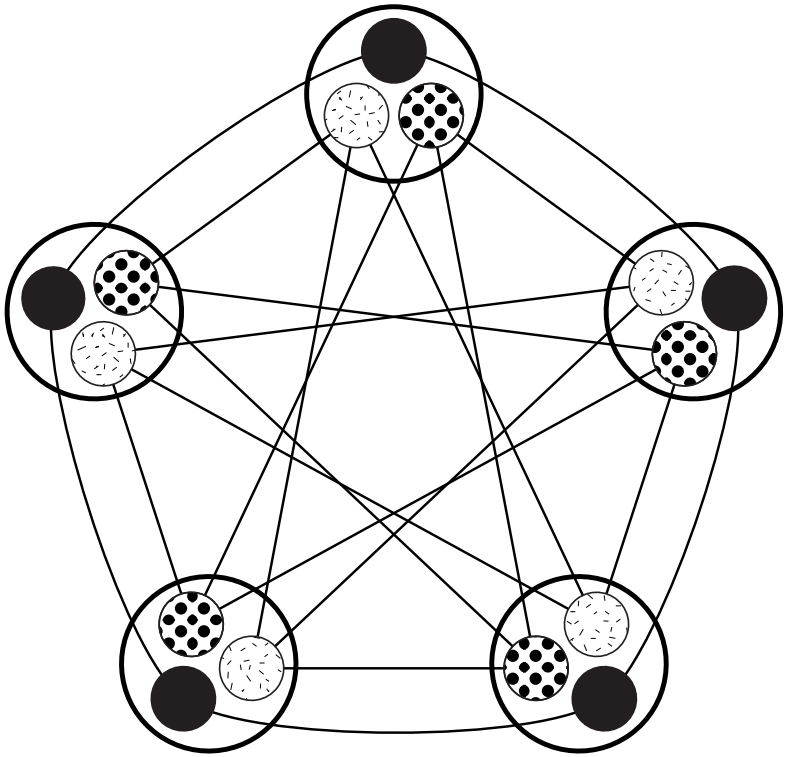
Egzemplarz:
Zbiór zmiennych V oraz zbiór C więzów postaci (v, a, u, b), gdzie $v, u \in V$ oraz $a, b \in \{1, 2, 3\}$.

Dla danego kolorowania $c : V \to \{1, 2, 3\}$ mówimy, że więz $(v, a, u, b) \in C$ jest **naruszony**, gdy $c(v) = a$ oraz $c(u) = b$; w przeciwnym przypadku jest **spełniony**.

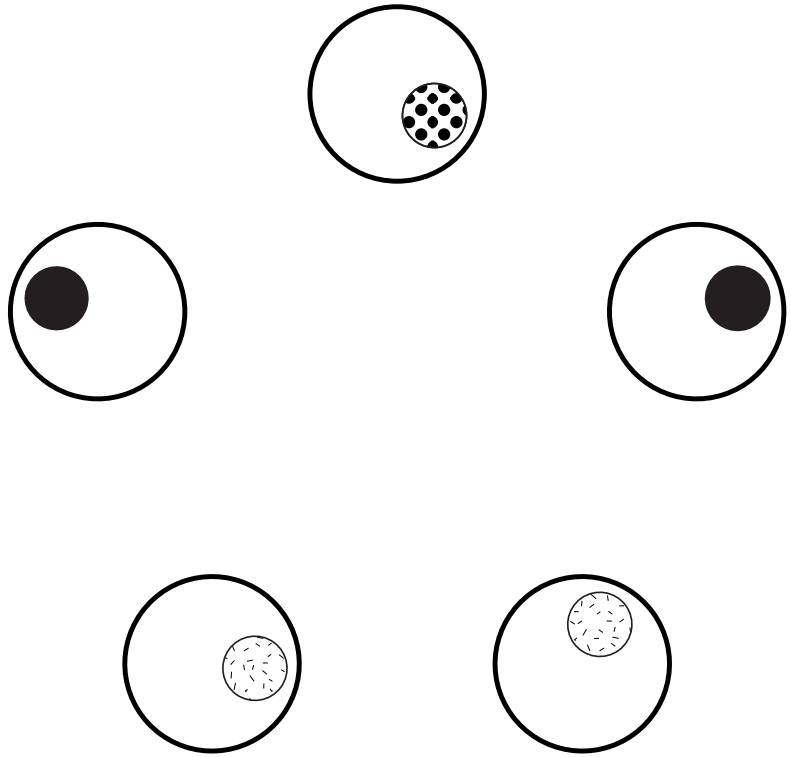
Problem:
Znaleźć kolorowanie $c : V \to \{1, 2, 3\}$ takie, wszystkie więzy z C są spełnione.

CSP = Constraint Satisfaction Problem

(3,2)-CSP: Przykład



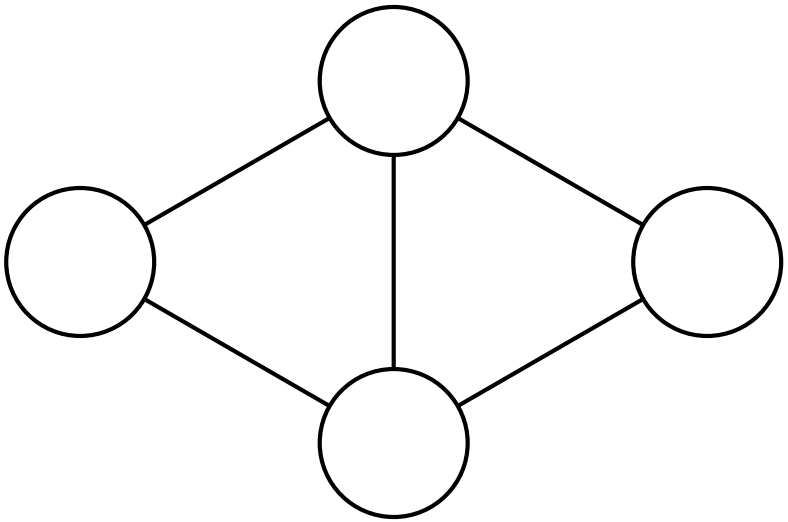
Egzemplarz



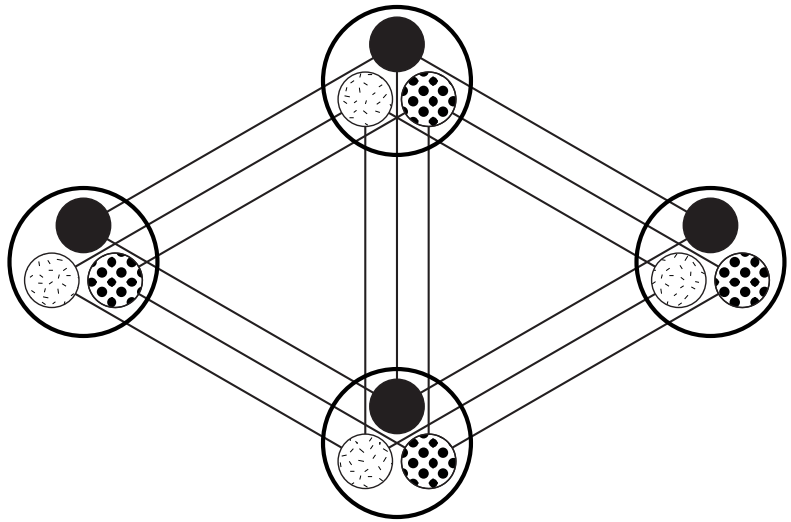
Rozwiązanie

(Ilustracje pochodzą z: Beigel, Eppstein 3-*coloring in time* $O(1.3289^n)$.)

3-kolorowanie jako (3,2)-CSP

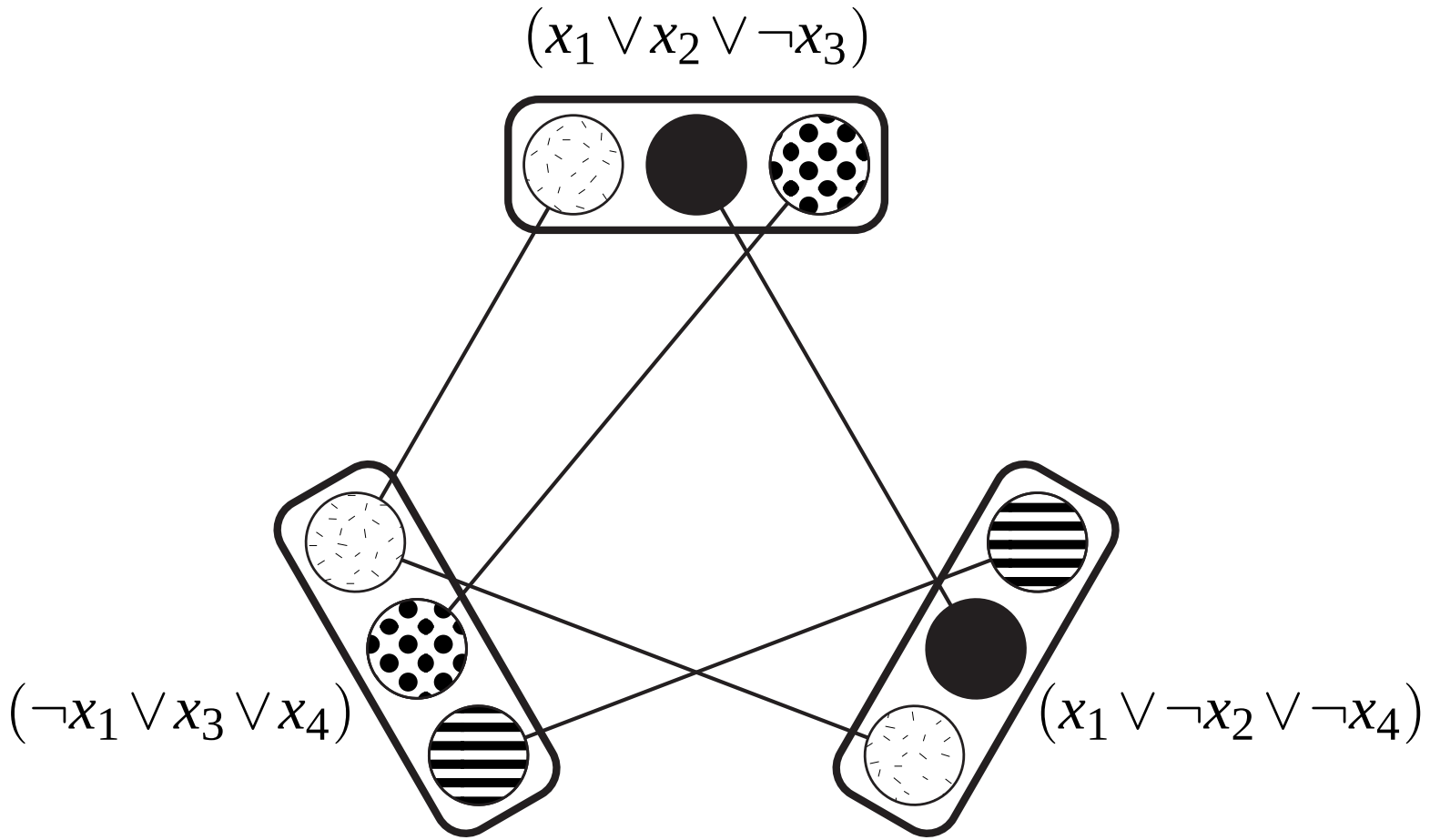


Egzemplarz 3-kolorowania



Egzemplarz $(3, 2)$-CSP.

3-SAT jako (3,2)-CSP

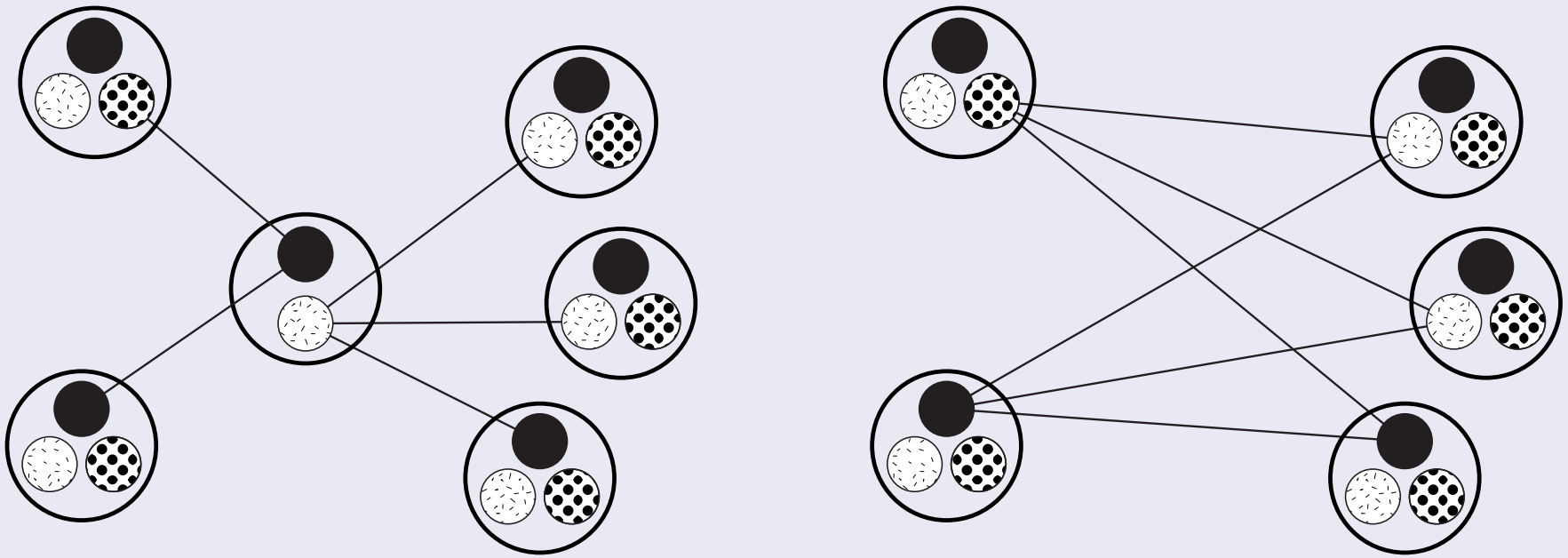


Redukowanie zmiennych z dwoma dozwolonymi kolorami

Lemat o dwóch kolorach

Jeśli egzemplarz $(3, 2)$-CSP zawiera zmienną, dla której są tylko 2 dozwolone kolory, to można znaleźć równoważny egzemplarz zawierający o jedną zmienną mniej.

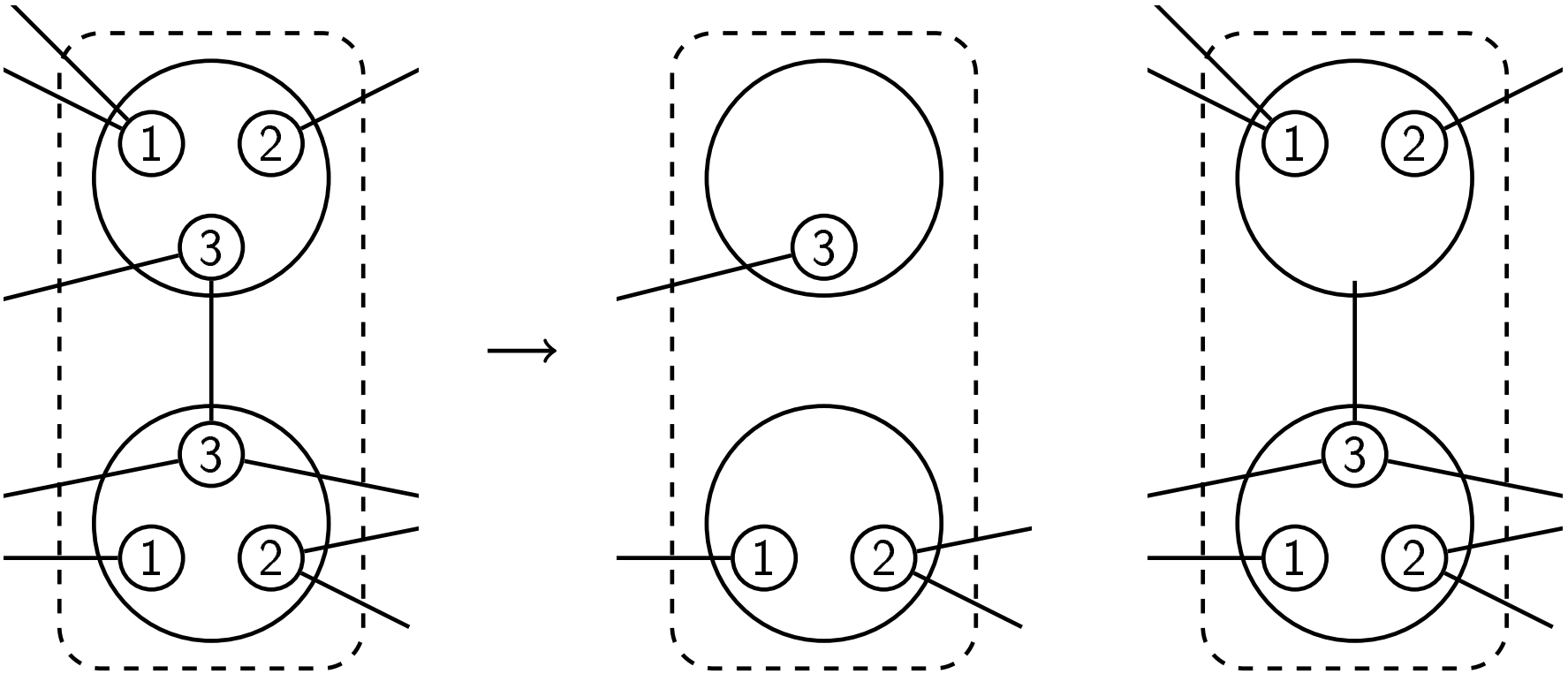
Dowód



Oryginalny egzemplarz $\rightarrow$ Nowy egzemplarz.

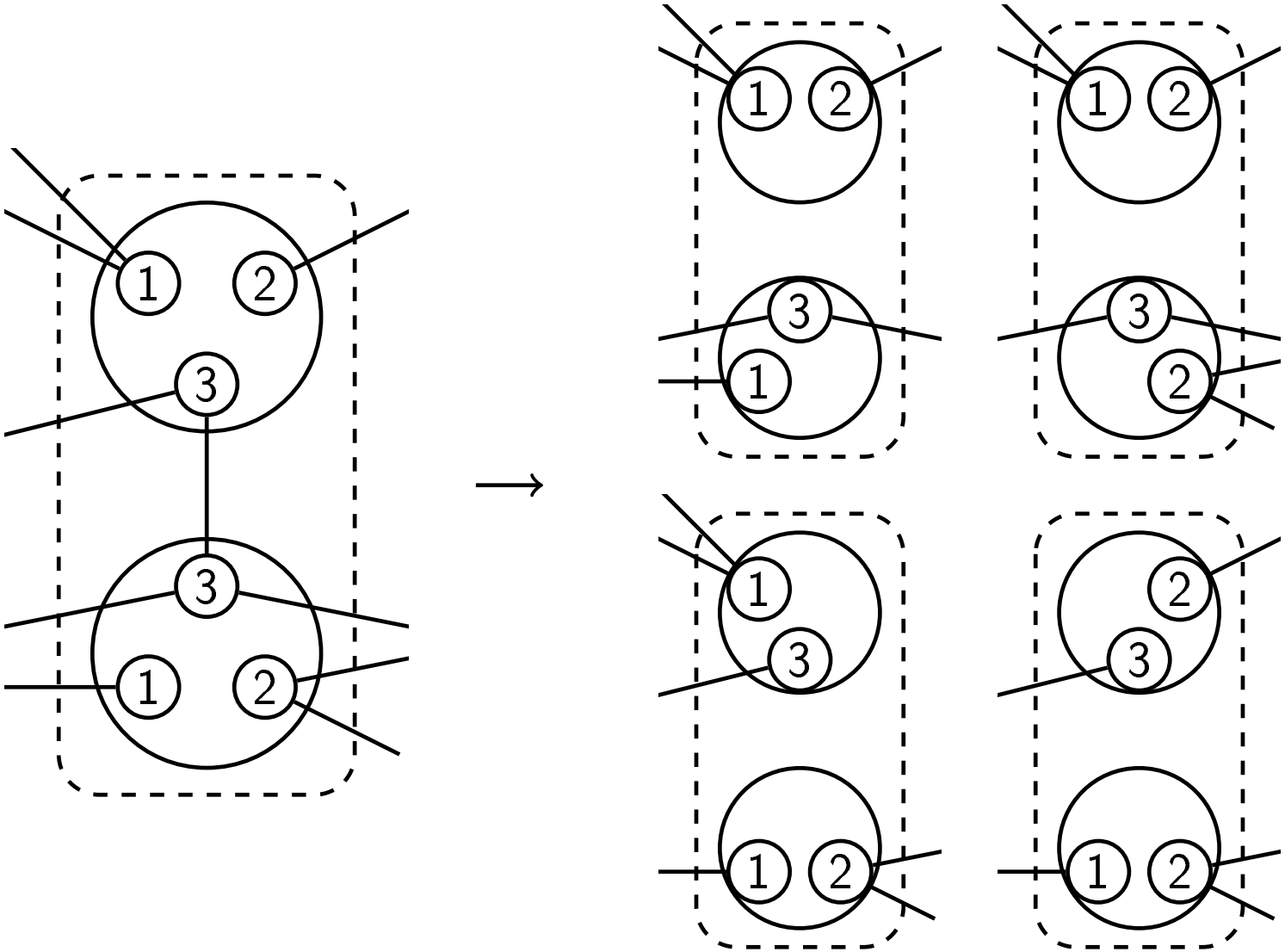
Nieciekawa reguła rozgałęziająca

Rozważmy dowolne dwa wierzchołki pojawiające się w pewnym więzie.



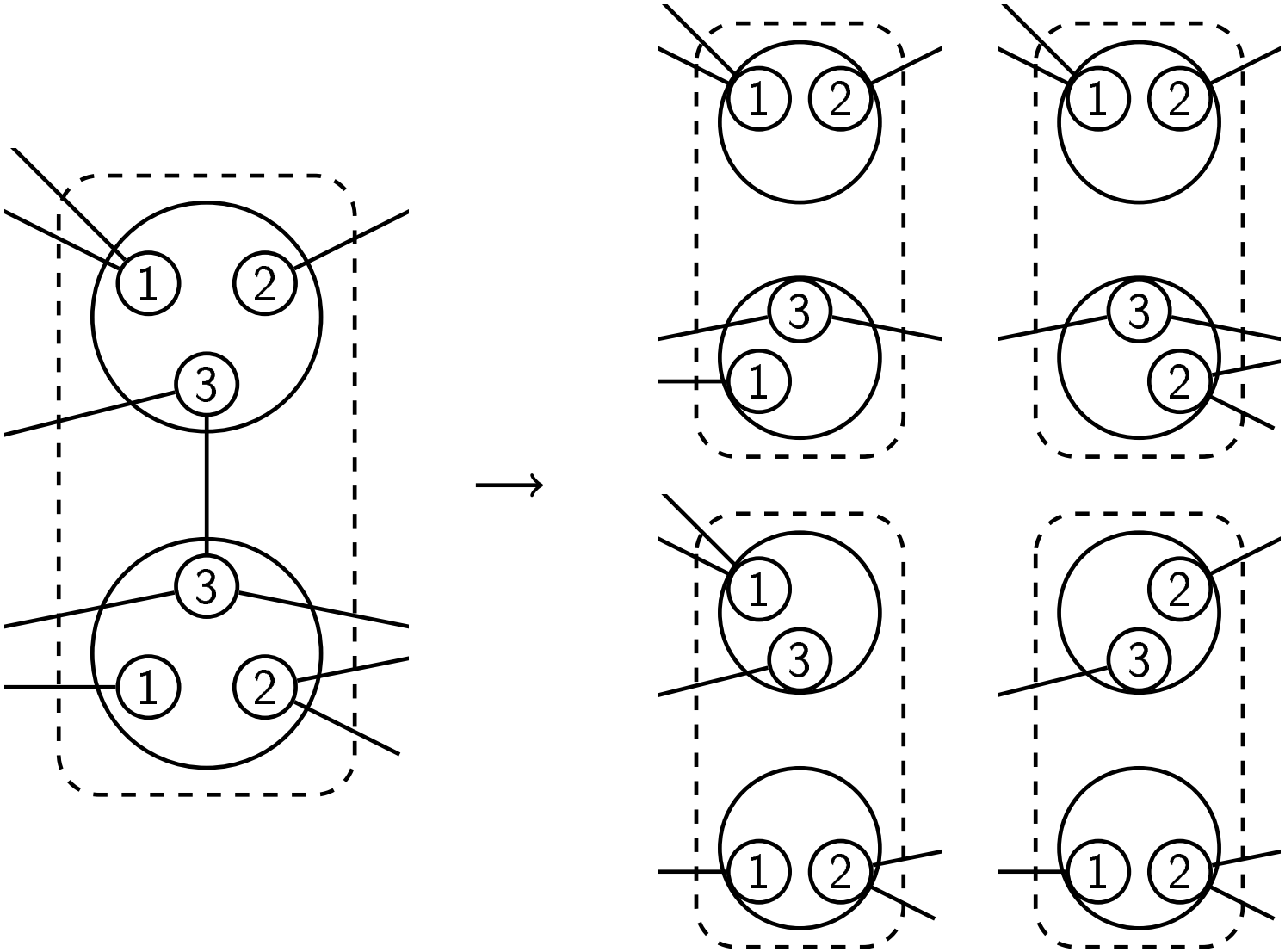
Z lematu o dwóch kolorach to daje wektor rozgałęzienia $(2, 1)$, czyli algorytm o złożoności $O(1.62^n)$.

Ciekawa reguła rozgałęziająca



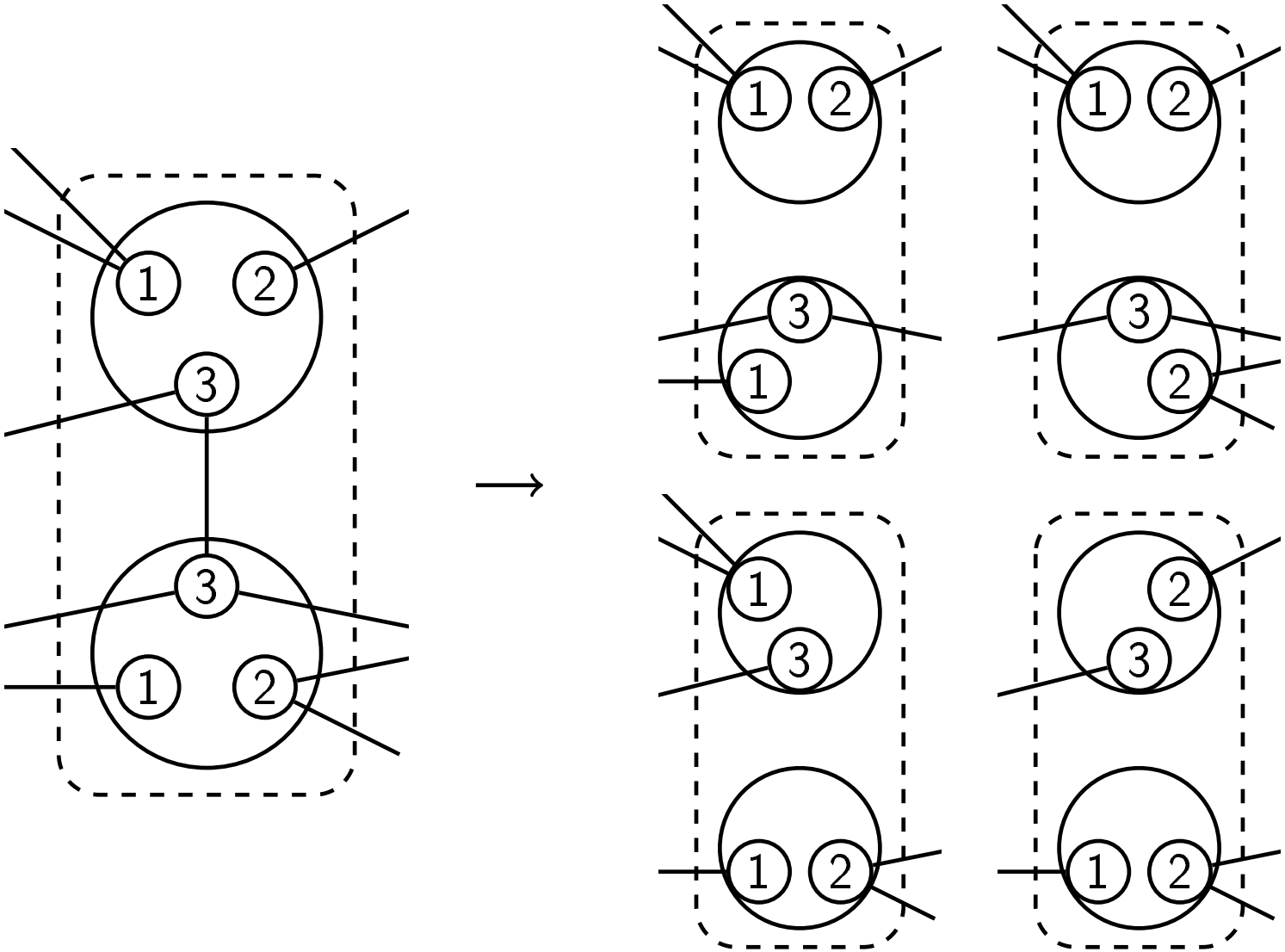
Z lematu o dwóch kolorach to daje wektor rozgałęzienia $(2, 2, 2, 2)$, czyli algorytm o złożoności $O(4^{n/2}) = O(2^n)$.

Ciekawa reguła rozgałęziająca



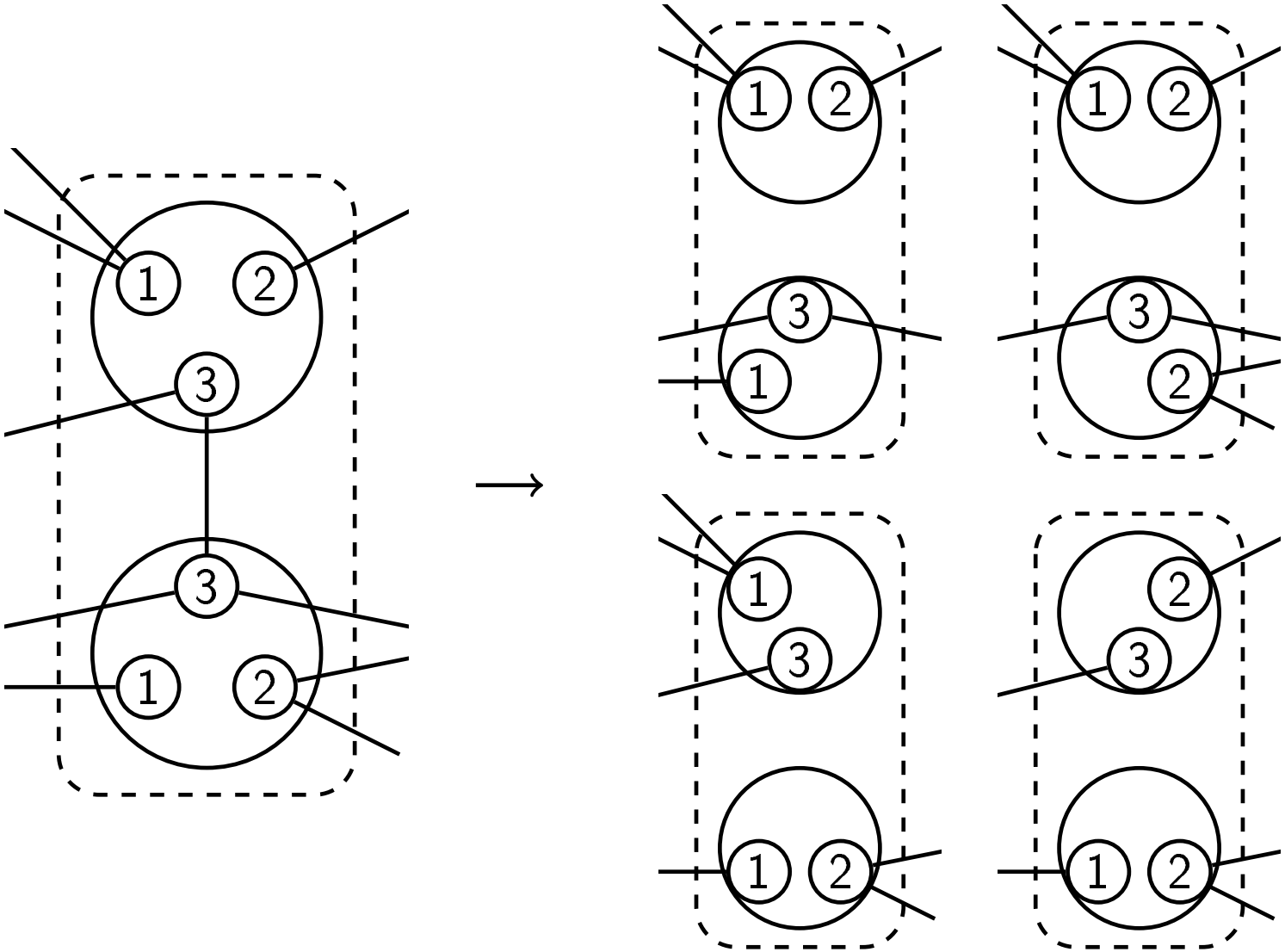
ale: jeśli istnieje rozwiązanie, to pojawia się ono w dokładnie **dwóch** z czterech gałęzi! Jak to wykorzystać?

Ciekawa reguła rozgałęziająca



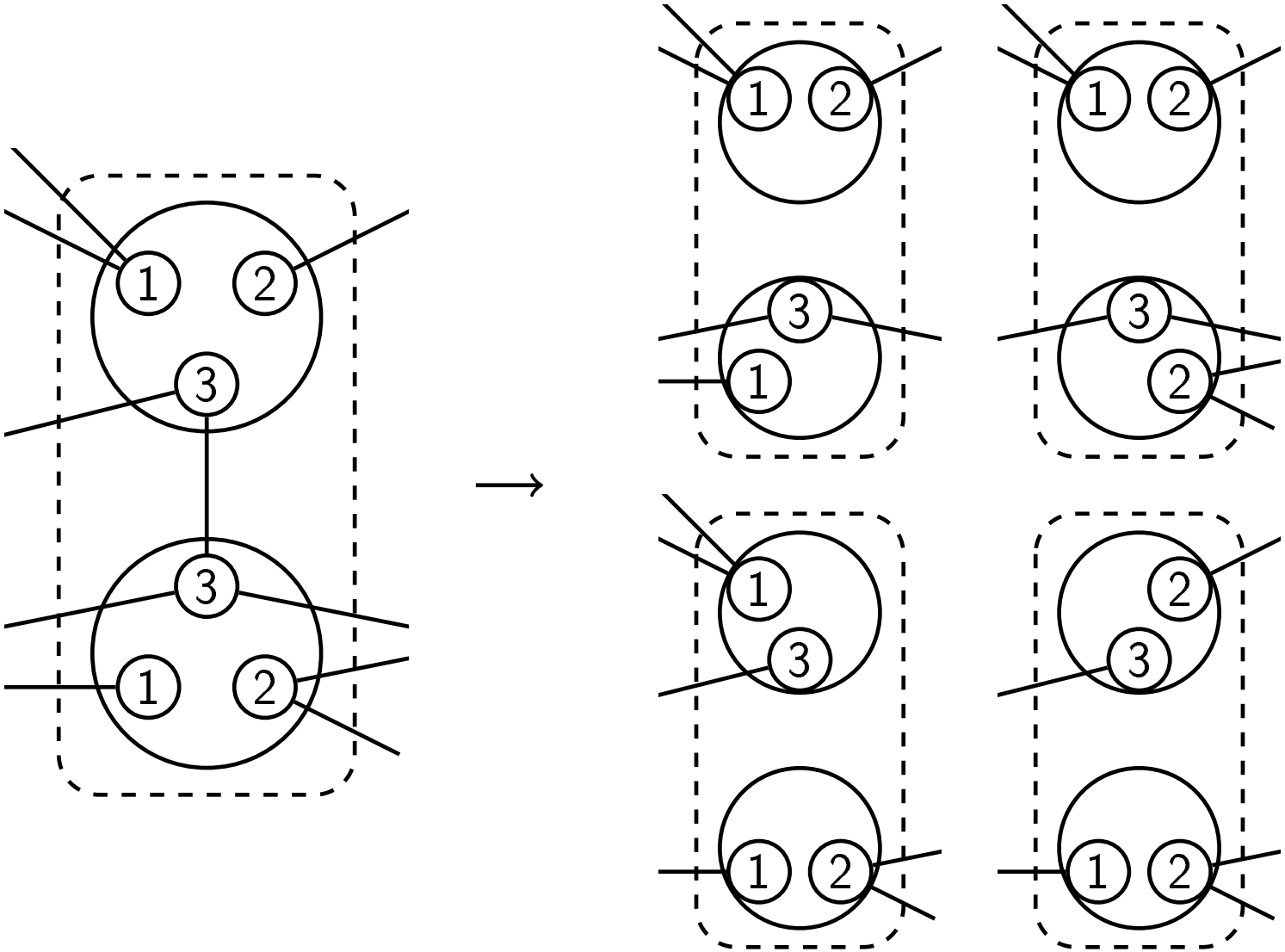
ale: jeśli istnieje rozwiązanie, to pojawia się ono w dokładnie **dwóch** z czterech gałęzi! Jak to wykorzystać? Randomizacja!

Ciekawa reguła rozgałęziająca



ale: jeśli istnieje rozwiązanie, to pojawia się ono w dokładnie **dwóch** z czterech gałęzi! Jak to wykorzystać? Randomizacja!

Ciekawa reguła rozgałęziająca



Z prawdopodobieństwem 1/4 wybieramy jeden z 4 mniejszych egzemplarzy.
Z prawdopodobieństwem 1/2 trafiamy.

$(3,2)$-CSP: Randomizowane rozgałęzianie

- Taka próba działa w czasie wielomianowym.
- Pr[Próba znajdzie rozwiązanie jeśli istnieje]$= (\frac{1}{2})^{n/2} = 2^{-n/2}$.

$(3,2)$-CSP: Randomizowane rozgałęzianie

- Taka próba działa w czasie wielomianowym.
- Pr[Próba znajdzie rozwiązanie jeśli istnieje]$= (\frac{1}{2})^{n/2} = 2^{-n/2}$.
- Pr[Próba **nie** znajdzie rozwiązania jeśli istnieje]$= 1 - 2^{-n/2}$.

$(3,2)$-CSP: Randomizowane rozgałęzianie

- Taka próba działa w czasie wielomianowym.
- Pr[Próba znajdzie rozwiązanie jeśli istnieje]$= (\frac{1}{2})^{n/2} = 2^{-n/2}$.
- Pr[Próba **nie** znajdzie rozwiązania jeśli istnieje]$= 1 - 2^{-n/2}$.
- Wykonujemy $100 \cdot 2^{n/2}$ prób. Jeśli żadna próba nie zwróciła rozwiązania, odpowiadamy, że rozwiązanie nie istnieje.

$(3,2)$-CSP: Randomizowane rozgałęzianie

- Taka próba działa w czasie wielomianowym.
- Pr[Próba znajdzie rozwiązanie jeśli istnieje]$= (\frac{1}{2})^{n/2} = 2^{-n/2}$.
- Pr[Próba **nie** znajdzie rozwiązania jeśli istnieje]$= 1 - 2^{-n/2}$.
- Wykonujemy $100 \cdot 2^{n/2}$ prób. Jeśli żadna próba nie zwróciła rozwiązania, odpowiadamy, że rozwiązanie nie istnieje.
- Pr[Żadna próba **nie** znajdzie rozwiązania jeśli istnieje] $= (1 - 2^{-n/2})^{100 \cdot 2^{n/2}} \leq e^{-100}$.

$(3,2)$-CSP: Randomizowane rozgałęzianie

- Taka próba działa w czasie wielomianowym.
- Pr[Próba znajdzie rozwiązanie jeśli istnieje]$= (\frac{1}{2})^{n/2} = 2^{-n/2}$.
- Pr[Próba **nie** znajdzie rozwiązania jeśli istnieje]$= 1 - 2^{-n/2}$.
- Wykonujemy $100 \cdot 2^{n/2}$ prób. Jeśli żadna próba nie zwróciła rozwiązania, odpowiadamy, że rozwiązanie nie istnieje.
- Pr[Żadna próba **nie** znajdzie rozwiązania jeśli istnieje] $= (1 - 2^{-n/2})^{100 \cdot 2^{n/2}} \leq e^{-100}$.
- Jeśli rozwiązanie nie istnieje, algorytm zawsze zwraca poprawną odpowiedź.

$(3,2)$-CSP: Randomizowane rozgałęzianie

- Taka próba działa w czasie wielomianowym.
- Pr[Próba znajdzie rozwiązanie jeśli istnieje]$= (\frac{1}{2})^{n/2} = 2^{-n/2}$.
- Pr[Próba **nie** znajdzie rozwiązania jeśli istnieje]$= 1 - 2^{-n/2}$.
- Wykonujemy $100 \cdot 2^{n/2}$ prób. Jeśli żadna próba nie zwróciła rozwiązania, odpowiadamy, że rozwiązanie nie istnieje.
- Pr[Żadna próba **nie** znajdzie rozwiązania jeśli istnieje] $= (1 - 2^{-n/2})^{100 \cdot 2^{n/2}} \leq e^{-100}$.
- Jeśli rozwiązanie nie istnieje, algorytm zawsze zwraca poprawną odpowiedź.

Wniosek [Beigel, Eppstein 2005]

Istnieje algorytm Monte-Carlo, który rozwiązuje problem $(3,2)$-CSP w czasie $O^*(2^{n/2})$ z dowolnie małym (jednostronnym) prawdopodobieństem błędu.

Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące

Oznaczenie: $\alpha(G) =$ rozmiar najliczniejszego zbioru niezależnego w G.

Dominowanie

Jeśli $N[w] \subseteq N[v]$ („v dominuje w") to $\alpha(G) = \alpha(G - v)$.

Dowód

W szczególności $vw \in E$. Jeśli v jest w pewnym najliczniejszym zbiorze niezależnym I to $w \notin I$, a więc $I - \{v\} \cup \{w\}$ jest zbiorem niezależnym o tej samej mocy. Czyli zawsze istnieje najliczniejszy zbiór niezależny, który nie zawiera v.

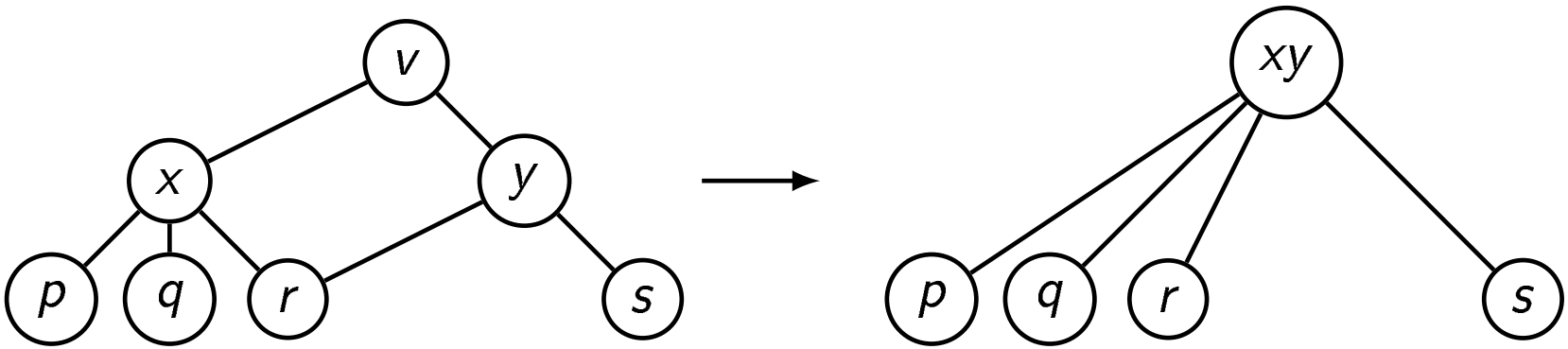
Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące

Zwijanie wierzchołków stopnia 2

Niech $\deg(v) = 2$, $N(v) = \{x, y\}$ i załóżmy, że v nie dominuje x ani y.
Niech G' będzie grafem, który powstaje z G przez:

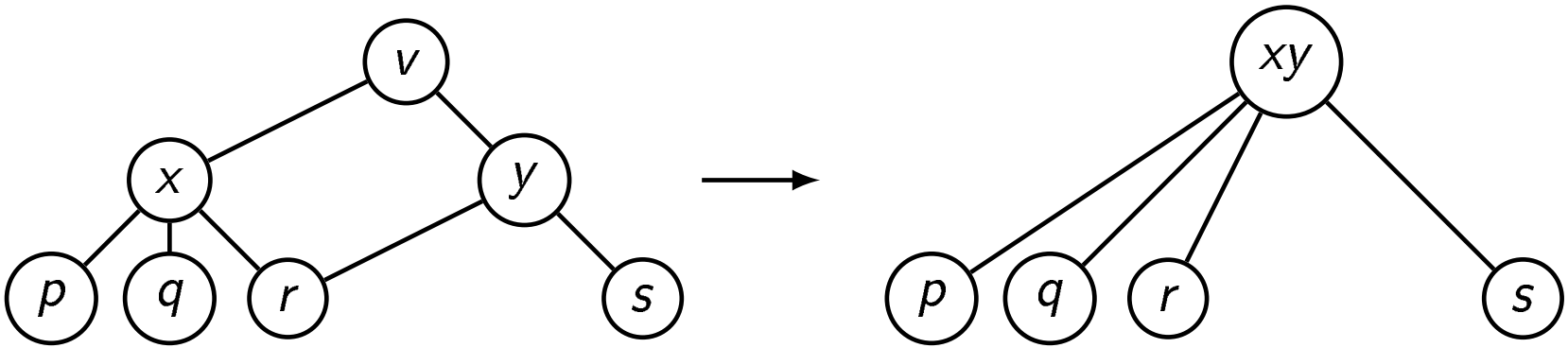
- usunięcie v, x, y,
- dodanie nowego wierzchołka xy,
- dodanie krawędzi od xy do każdego $w \in (N_G(x) \cup N_G(y)) \setminus \{v\}$

Wtedy $\alpha(G) = 1 + \alpha(G')$



Dowód: Zauważmy, że $xy \notin E(G)$.

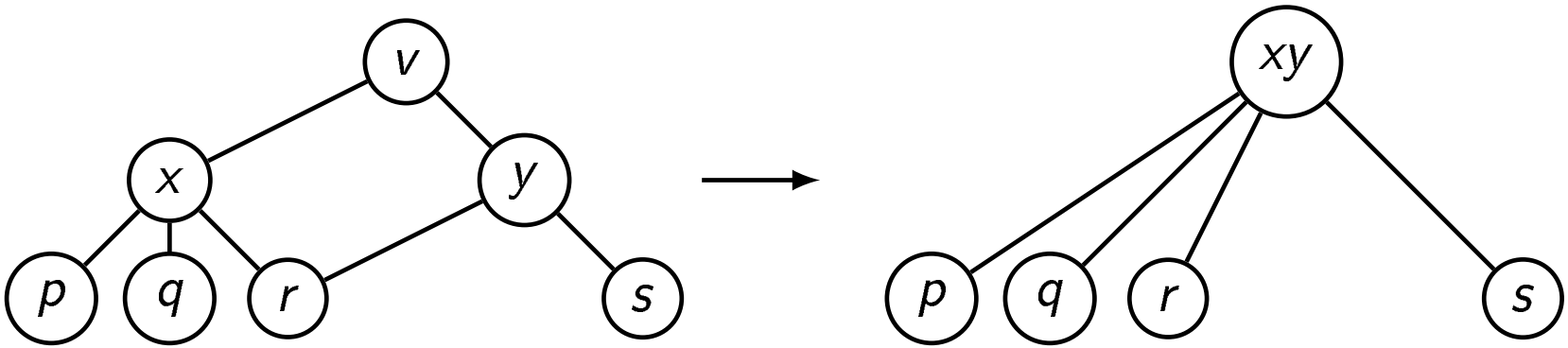
Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące



Dowód: Zauważmy, że $xy \notin E(G)$.
Niech I' będzie najliczniejszym zbiorem niezależnym w G'.

- Jeśli $xy \in I'$ to $I' \setminus \{xy\} \cup \{x, y\}$ jest niezależny w G.

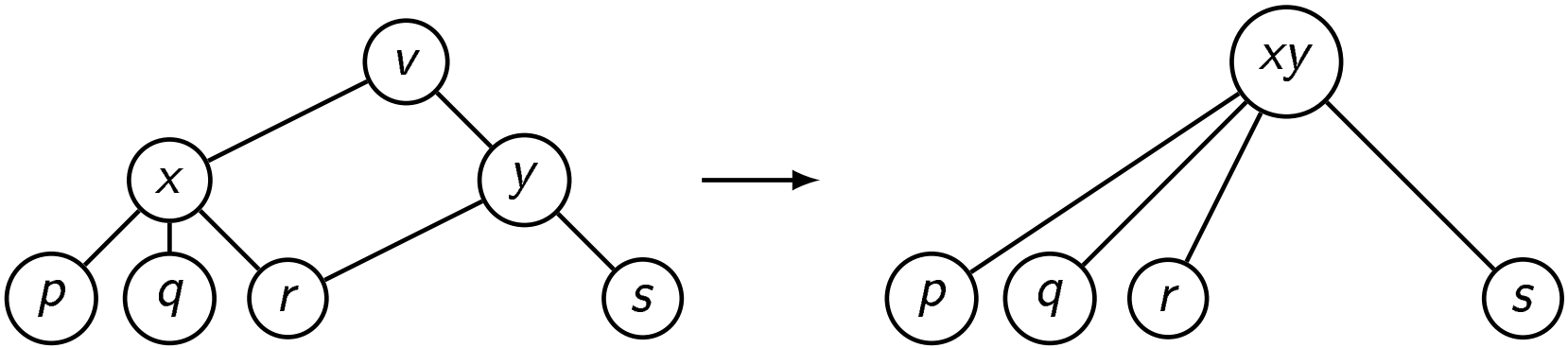
Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące



Dowód: Zauważmy, że $xy \notin E(G)$.
Niech I' będzie najliczniejszym zbiorem niezależnym w G'.

- Jeśli $xy \in I'$ to $I' \setminus \{xy\} \cup \{x, y\}$ jest niezależny w G.
- Jeśli $xy \notin I'$ to $I' \cup \{v\}$ jest niezależny w G.

Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące



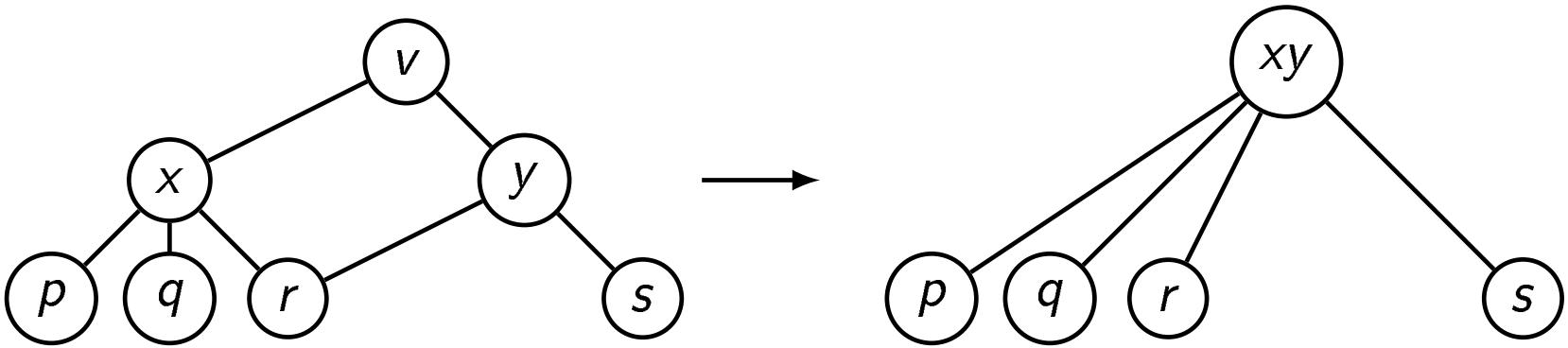
Dowód: Zauważmy, że $xy \notin E(G)$.
Niech I' będzie najliczniejszym zbiorem niezależnym w G'.

- Jeśli $xy \in I'$ to $I' \setminus \{xy\} \cup \{x, y\}$ jest niezależny w G.
- Jeśli $xy \notin I'$ to $I' \cup \{v\}$ jest niezależny w G.
- Stąd, $\alpha(G) \geq \alpha(G') - 1$.

Niech I będzie najliczniejszym zbiorem niezależnym w G.

- Jeśli $x, y \in I$ to $I \cup \{x, y\}$ jest niezależny w G'.

Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące



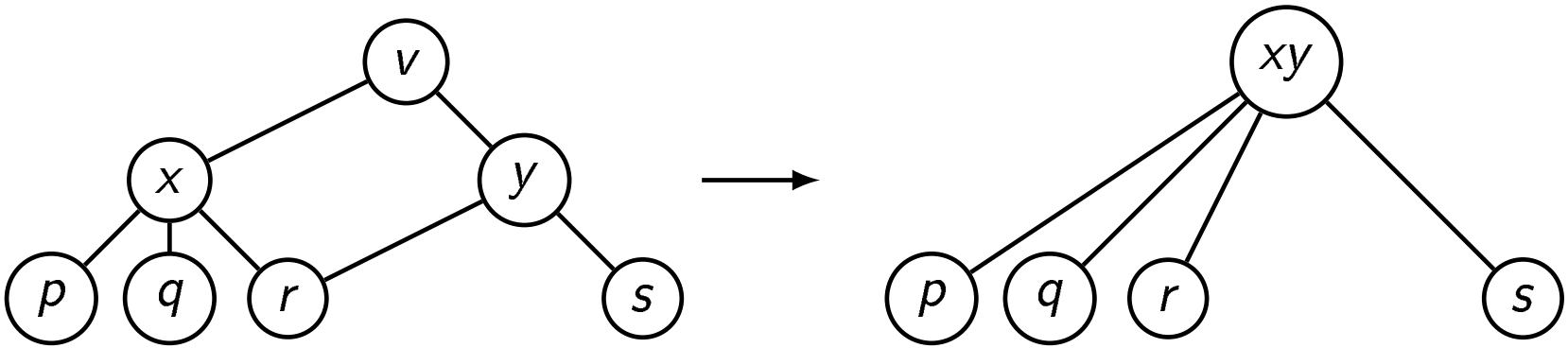
Dowód: Zauważmy, że $xy \notin E(G)$.
Niech I' będzie najliczniejszym zbiorem niezależnym w G'.

- Jeśli $xy \in I'$ to $I' \setminus \{xy\} \cup \{x, y\}$ jest niezależny w G.
- Jeśli $xy \notin I'$ to $I' \cup \{v\}$ jest niezależny w G.
- Stąd, $\alpha(G) \geq \alpha(G') - 1$.

Niech I będzie najliczniejszym zbiorem niezależnym w G.

- Jeśli $x, y \in I$ to $I \cup \{x, y\}$ jest niezależny w G'.
- Jeśli $x \notin I$ lub $y \notin I$ to bez straty ogólności $v \in I$, $x, y \notin I$. Wtedy $I \setminus \{v\}$ jest niezależny w G'.

Najliczniejszy zbiór niezależny: dwie nowe reguły redukujące



Dowód: Zauważmy, że $xy \notin E(G)$.
Niech I' będzie najliczniejszym zbiorem niezależnym w G'.

- Jeśli $xy \in I'$ to $I' \setminus \{xy\} \cup \{x, y\}$ jest niezależny w G.
- Jeśli $xy \notin I'$ to $I' \cup \{v\}$ jest niezależny w G.
- Stąd, $\alpha(G) \geq \alpha(G') - 1$.

Niech I będzie najliczniejszym zbiorem niezależnym w G.

- Jeśli $x, y \in I$ to $I \cup \{x, y\}$ jest niezależny w G'.
- Jeśli $x \notin I$ lub $y \notin I$ to bez straty ogólności $v \in I$, $x, y \notin I$. Wtedy $I \setminus \{v\}$ jest niezależny w G'.
- Stąd, $\alpha(G') \geq \alpha(G) + 1$, czyli $\alpha(G) \leq \alpha(G') - 1$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Algorytm przez rozgałęzianie, v4.0

procedure MIS(G)

1: **if** $|V(G)| \leq 1$ **then**
2: **return** $|V(G)|$
3: **if** $\exists v, w \ : \ N[w] \subseteq N[v]$ **then**
4: **return** MIS($G - v$)
5: **else if** $\exists v \ : \ \deg(v) \leq 1$ **then**
6: **return** $1 +$ MIS($G - N[v]$)
7: **else if** $\exists v \ : \ \deg(v) = 2$ **then**
8: $G' \leftarrow$ graf po zwinięciu v,
9: **return** $1 +$ MIS(G')
10: $v \leftarrow$ wierzchołek maksymalnego stopnia w G.
11: **return** $\max\{\text{MIS}(G - v), 1 + \text{MIS}(G - N[v])\}$

Przypomnijmy: złożoność czasowa to $O^*(\lambda(1,4)^n) = O(1.381^n)$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma”.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma”.

Nowa analiza tego samego algorytmu

- Pomysł:
  - Wierzchołki stopnia $\deg(x) \leq 2$ będą miały wagę $w(x) = 0$,

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma”.

Nowa analiza tego samego algorytmu

- Pomysł:
  - Wierzchołki stopnia $\deg(x) \leq 2$ będą miały wagę $w(x) = 0$,
  - wierzchołki stopnia $\deg(x) \geq 3$ będą miały wagę $w(x) = 1$,

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma".

Nowa analiza tego samego algorytmu

- Pomysł:
  - Wierzchołki stopnia $\deg(x) \leq 2$ będą miały wagę $w(x) = 0$,
  - wierzchołki stopnia $\deg(x) \geq 3$ będą miały wagę $w(x) = 1$,
  - wtedy rozmiar egzemplarza $\mu(G) = \sum_{v \in V} w(v)$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma".

Nowa analiza tego samego algorytmu

- Pomysł:
  - Wierzchołki stopnia $\deg(x) \leq 2$ będą miały wagę $w(x) = 0$,
  - wierzchołki stopnia $\deg(x) \geq 3$ będą miały wagę $w(x) = 1$,
  - wtedy rozmiar egzemplarza $\mu(G) = \sum_{v \in V} w(v)$.
- dla $\deg(v) = 3$:
  - Wszsytkie wierzchołki w grafie mają stopień 3! A więc:
  - W egzemplarzu $G - v$ pojawią się 3 wierzchołki stopnia ≤ 2.

Czyli mamy czynnik pracy $\lambda(4,4) = 2^{1/4} = 1.19$ (a nawet lepszy...)

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma".

Nowa analiza tego samego algorytmu

- Pomysł:
  - Wierzchołki stopnia $\deg(x) \leq 2$ będą miały wagę $w(x) = 0$,
  - wierzchołki stopnia $\deg(x) \geq 3$ będą miały wagę $w(x) = 1$,
  - wtedy rozmiar egzemplarza $\mu(G) = \sum_{v \in V} w(v)$.
- dla $\deg(v) = 3$:
  - Wszsytkie wierzchołki w grafie mają stopień 3! A więc:
  - W egzemplarzu $G - v$ pojawią się 3 wierzchołki stopnia ≤ 2.

  Czyli mamy czynnik pracy $\lambda(4,4) = 2^{1/4} = 1.19$ (a nawet lepszy...)
- dla $\deg(v) \geq 4$: mamy czynnik pracy $\lambda(1,5) = 1.33$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Obserwacja

Wierzchołki stopnia ≤ 2 znikają w czasie wielomianowym. W jakiś sensie „już ich nie ma".

Nowa analiza tego samego algorytmu

- Pomysł:
  - Wierzchołki stopnia $\deg(x) \leq 2$ będą miały wagę $w(x) = 0$,
  - wierzchołki stopnia $\deg(x) \geq 3$ będą miały wagę $w(x) = 1$,
  - wtedy rozmiar egzemplarza $\mu(G) = \sum_{v \in V} w(v)$.
- dla $\deg(v) = 3$:
  - Wszsytkie wierzchołki w grafie mają stopień 3! A więc:
  - W egzemplarzu $G - v$ pojawią się 3 wierzchołki stopnia ≤ 2.

  Czyli mamy czynnik pracy $\lambda(4,4) = 2^{1/4} = 1.19$ (a nawet lepszy...)
- dla $\deg(v) \geq 4$: mamy czynnik pracy $\lambda(1,5) = 1.33$.
- **Wniosek:** Czas $O(1.33^{\mu(G)}) = O(1.33^n)$, bo $\mu(G) \leq n$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Co tu się stało? Wprowadziliśmy **nową miarę rozmiaru egzemplarza**, która pozwoliła lepiej mierzyć postęp, jaki wykonuje algorytm w jednej z gałęzi (a nawet obu) w przypadku gdy $\deg(v) = 3$, który był „wąskim gardłem".

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Co tu się stało? Wprowadziliśmy **nową miarę rozmiaru egzemplarza**, która pozwoliła lepiej mierzyć postęp, jaki wykonuje algorytm w jednej z gałęzi (a nawet obu) w przypadku gdy $\deg(v) = 3$, który był „wąskim gardłem".
- Idźmy dalej tym tropem. Aktualne wąskie gardło: wierzchołek v stopnia $\deg(v) = 4$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Co tu się stało? Wprowadziliśmy **nową miarę rozmiaru egzemplarza**, która pozwoliła lepiej mierzyć postęp, jaki wykonuje algorytm w jednej z gałęzi (a nawet obu) w przypadku gdy $\deg(v) = 3$, który był „wąskim gardłem".
- Idźmy dalej tym tropem. Aktualne wąskie gardło: wierzchołek v stopnia $\deg(v) = 4$.
  - Każdy wierzchołek w grafie stopnia 3 lub 4.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Co tu się stało? Wprowadziliśmy **nową miarę rozmiaru egzemplarza**, która pozwoliła lepiej mierzyć postęp, jaki wykonuje algorytm w jednej z gałęzi (a nawet obu) w przypadku gdy $\deg(v) = 3$, który był „wąskim gardłem”.
- Idźmy dalej tym tropem. Aktualne wąskie gardło: wierzchołek v stopnia $\deg(v) = 4$.
  - Każdy wierzchołek w grafie stopnia 3 lub 4.
  - Wąskie gardło: v ma 4 sąsiadów stopnia 4; w egzemplarzu $G - v$ ich stopnie maleją do 3.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Co tu się stało? Wprowadziliśmy **nową miarę rozmiaru egzemplarza**, która pozwoliła lepiej mierzyć postęp, jaki wykonuje algorytm w jednej z gałęzi (a nawet obu) w przypadku gdy $\deg(v) = 3$, który był „wąskim gardłem".
- Idźmy dalej tym tropem. Aktualne wąskie gardło: wierzchołek v stopnia $\deg(v) = 4$.
  - Każdy wierzchołek w grafie stopnia 3 lub 4.
  - Wąskie gardło: v ma 4 sąsiadów stopnia 4; w egzemplarzu $G - v$ ich stopnie maleją do 3.
  - **Obserwacja:** Wierzchołek stopnia 3 już niedługo może stać się stopnia 2 i zniknąć! Powinniśmy to uwzględnić w $\mu(G)$: takie wierzchołki powinny mieć wagę $\alpha \in (0, 1)$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Co tu się stało? Wprowadziliśmy **nową miarę rozmiaru egzemplarza**, która pozwoliła lepiej mierzyć postęp, jaki wykonuje algorytm w jednej z gałęzi (a nawet obu) w przypadku gdy $\deg(v) = 3$, który był „wąskim gardłem".
- Idźmy dalej tym tropem. Aktualne wąskie gardło: wierzchołek v stopnia $\deg(v) = 4$.
  - Każdy wierzchołek w grafie stopnia 3 lub 4.
  - Wąskie gardło: v ma 4 sąsiadów stopnia 4; w egzemplarzu $G - v$ ich stopnie maleją do 3.
  - **Obserwacja:** Wierzchołek stopnia 3 już niedługo może stać się stopnia 2 i zniknąć! Powinniśmy to uwzględnić w $\mu(G)$: takie wierzchołki powinny mieć wagę $\alpha \in (0, 1)$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Dostajemy:
  - dla $\deg(v) = 3$ czynnik pracy $\lambda(4\alpha, 4\alpha) = 2^{1/(4\alpha)}$
  - dla $\deg(v) = 4$ czynnik pracy $\lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha)$
  - dla $\deg(v) \geq 5$ czynnik pracy $\lambda(1 + \alpha\#_3, 1 + \alpha\#_3 + (\deg(v) - \#_3)) \leq \lambda(1, 6) = 1.2852$, gdzie $\#_3$ oznacza liczbę sąsiadów v stopnia 3, nierówność zachodzi dla $\alpha \geq \frac{1}{2}$, ponieważ $\lambda(a, b) \leq \lambda(c, d)$ dla $a + b = c + d$, $\min\{a, b\} \geq \min\{c, d\}$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Dostajemy:
  - dla $\deg(v) = 3$ czynnik pracy $\lambda(4\alpha, 4\alpha) = 2^{1/(4\alpha)}$
  - dla $\deg(v) = 4$ czynnik pracy $\lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha)$
  - dla $\deg(v) \geq 5$ czynnik pracy $\lambda(1 + \alpha\#_3, 1 + \alpha\#_3 + (\deg(v) - \#_3)) \leq \lambda(1, 6) = 1.2852$, gdzie $\#_3$ oznacza liczbę sąsiadów v stopnia 3, nierówność zachodzi dla $\alpha \geq \frac{1}{2}$, ponieważ $\lambda(a, b) \leq \lambda(c, d)$ dla $a + b = c + d$, $\min\{a, b\} \geq \min\{c, d\}$.
- Aby zwijanie wierzchołka v o sąsiadach x i y nie zwiększało rozmiaru egzemplarza, chcemy, żeby $w(xy) \leq w(v) + w(x) + w(y)$. Ponieważ $w(xy) \leq 1$ oraz $2\alpha \leq w(v) + w(x) + w(y)$, wystarczy, żeby $1 \leq 2\alpha$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Dostajemy:
  - dla $\deg(v) = 3$ czynnik pracy $\lambda(4\alpha, 4\alpha) = 2^{1/(4\alpha)}$
  - dla $\deg(v) = 4$ czynnik pracy $\lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha)$
  - dla $\deg(v) \geq 5$ czynnik pracy $\lambda(1 + \alpha\#_3, 1 + \alpha\#_3 + (\deg(v) - \#_3)) \leq \lambda(1, 6) = 1.2852$, gdzie $\#_3$ oznacza liczbę sąsiadów v stopnia 3, nierówność zachodzi dla $\alpha \geq \frac{1}{2}$, ponieważ $\lambda(a, b) \leq \lambda(c, d)$ dla $a + b = c + d$, $\min\{a, b\} \geq \min\{c, d\}$.
- Aby zwijanie wierzchołka v o sąsiadach x i y nie zwiększało rozmiaru egzemplarza, chcemy, żeby $w(xy) \leq w(v) + w(x) + w(y)$. Ponieważ $w(xy) \leq 1$ oraz $2\alpha \leq w(v) + w(x) + w(y)$, wystarczy, żeby $1 \leq 2\alpha$.
- Obliczamy czynniki pracy dla wszystkich $\alpha \in (0, 1)$ z krokiem np 0.0001.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Dostajemy:
  - dla $\deg(v) = 3$ czynnik pracy $\lambda(4\alpha, 4\alpha) = 2^{1/(4\alpha)}$
  - dla $\deg(v) = 4$ czynnik pracy $\lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha)$
  - dla $\deg(v) \geq 5$ czynnik pracy $\lambda(1 + \alpha\#_3, 1 + \alpha\#_3 + (\deg(v) - \#_3)) \leq \lambda(1, 6) = 1.2852$, gdzie $\#_3$ oznacza liczbę sąsiadów v stopnia 3, nierówność zachodzi dla $\alpha \geq \frac{1}{2}$, ponieważ $\lambda(a, b) \leq \lambda(c, d)$ dla $a + b = c + d$, $\min\{a, b\} \geq \min\{c, d\}$.
- Aby zwijanie wierzchołka v o sąsiadach x i y nie zwiększało rozmiaru egzemplarza, chcemy, żeby $w(xy) \leq w(v) + w(x) + w(y)$. Ponieważ $w(xy) \leq 1$ oraz $2\alpha \leq w(v) + w(x) + w(y)$, wystarczy, żeby $1 \leq 2\alpha$.
- Obliczamy czynniki pracy dla wszystkich $\alpha \in (0, 1)$ z krokiem np 0.0001.
- Wychodzi, że dla $\alpha = 0.7259$ mamy $\lambda(4\alpha, 4\alpha) \approx \lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha) \approx 1.2697$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Dostajemy:
  - dla $\deg(v) = 3$ czynnik pracy $\lambda(4\alpha, 4\alpha) = 2^{1/(4\alpha)}$
  - dla $\deg(v) = 4$ czynnik pracy $\lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha)$
  - dla $\deg(v) \geq 5$ czynnik pracy $\lambda(1 + \alpha\#_3, 1 + \alpha\#_3 + (\deg(v) - \#_3)) \leq \lambda(1, 6) = 1.2852$, gdzie $\#_3$ oznacza liczbę sąsiadów v stopnia 3, nierówność zachodzi dla $\alpha \geq \frac{1}{2}$, ponieważ $\lambda(a, b) \leq \lambda(c, d)$ dla $a + b = c + d$, $\min\{a, b\} \geq \min\{c, d\}$.
- Aby zwijanie wierzchołka v o sąsiadach x i y nie zwiększało rozmiaru egzemplarza, chcemy, żeby $w(xy) \leq w(v) + w(x) + w(y)$. Ponieważ $w(xy) \leq 1$ oraz $2\alpha \leq w(v) + w(x) + w(y)$, wystarczy, żeby $1 \leq 2\alpha$.
- Obliczamy czynniki pracy dla wszystkich $\alpha \in (0, 1)$ z krokiem np 0.0001.
- Wychodzi, że dla $\alpha = 0.7259$ mamy $\lambda(4\alpha, 4\alpha) \approx \lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha) \approx 1.2697$.
- **Wniosek:** czas $O(1.2852^n)$.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Dostajemy:
  - dla $\deg(v) = 3$ czynnik pracy $\lambda(4\alpha, 4\alpha) = 2^{1/(4\alpha)}$
  - dla $\deg(v) = 4$ czynnik pracy $\lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha)$
  - dla $\deg(v) \geq 5$ czynnik pracy $\lambda(1 + \alpha\#_3, 1 + \alpha\#_3 + (\deg(v) - \#_3)) \leq \lambda(1, 6) = 1.2852$, gdzie $\#_3$ oznacza liczbę sąsiadów v stopnia 3, nierówność zachodzi dla $\alpha \geq \frac{1}{2}$, ponieważ $\lambda(a, b) \leq \lambda(c, d)$ dla $a + b = c + d$, $\min\{a, b\} \geq \min\{c, d\}$.
- Aby zwijanie wierzchołka v o sąsiadach x i y nie zwiększało rozmiaru egzemplarza, chcemy, żeby $w(xy) \leq w(v) + w(x) + w(y)$. Ponieważ $w(xy) \leq 1$ oraz $2\alpha \leq w(v) + w(x) + w(y)$, wystarczy, żeby $1 \leq 2\alpha$.
- Obliczamy czynniki pracy dla wszystkich $\alpha \in (0, 1)$ z krokiem np 0.0001.
- Wychodzi, że dla $\alpha = 0.7259$ mamy $\lambda(4\alpha, 4\alpha) \approx \lambda(1 + 4\min\{\alpha, 1-\alpha\}, 1 + 4\alpha) \approx 1.2697$.
- **Wniosek:** czas $O(1.2852^n)$.
- **Co tu się stało?**
  - Jeszcze lepiej mierzymy postęp algorytmu,
  - równoważymy czynniki pracy.

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Idziemy dalej: w_d = waga wierzchołka stopnia d.
- $0 = w_0 = w_1 = w_2 < w_3 \leq w_4 \leq w_5 \leq w_6 \leq w_{7+} = 1$.
- Nie rozważamy juz $w_7, w_8, \ldots$ osobno bo nie tam będzie wąskie gardło.
- Dla wierzchołka stopnia d czynnik pracy $\lambda(w_d + d \min_{k \leq d}(w_k - w_{k-1}), w_d + d \cdot w_3)$.
- Albo dokładniej: dla wierzchołka v stopnia d, który...
  - ... ma samych sąsiadów stopnia 3, czynnik pracy $\lambda(w_d + d \cdot w_3, w_d + d \cdot w_3 + [\text{sąsiedzi sąsiadów}])$.
  - ... ma samych sąsiadów stopnia d, czynnik pracy $\lambda(w_d + d \cdot (w_d - w_{d-1}), w_d + d \cdot w_d + [\text{sąsiedzi sąsiadów}])$.
  - ... itd dla wszystkich możliwości sąsiedztwa v (jest ich kilka tysięcy).
- **Problem:** Jak optymalizować wagi gdy jest ich tak dużo?

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Idziemy dalej: w_d = waga wierzchołka stopnia d.
- $0 = w_0 = w_1 = w_2 < w_3 \leq w_4 \leq w_5 \leq w_6 \leq w_{7+} = 1$.
- Nie rozważamy juz $w_7, w_8, \ldots$ osobno bo nie tam będzie wąskie gardło.
- Dla wierzchołka stopnia d czynnik pracy $\lambda(w_d + d \min_{k \leq d}(w_k - w_{k-1}), w_d + d \cdot w_3)$.
- Albo dokładniej: dla wierzchołka v stopnia d, który...
  - ... ma samych sąsiadów stopnia 3, czynnik pracy $\lambda(w_d + d \cdot w_3, w_d + d \cdot w_3 + [\text{sąsiedzi sąsiadów}])$.
  - ... ma samych sąsiadów stopnia d, czynnik pracy $\lambda(w_d + d \cdot (w_d - w_{d-1}), w_d + d \cdot w_d + [\text{sąsiedzi sąsiadów}])$.
  - ... itd dla wszystkich możliwości sąsiedztwa v (jest ich kilka tysięcy).
- **Problem:** Jak optymalizować wagi gdy jest ich tak dużo?
  - local search (simulated annealing),

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

- Idziemy dalej: w_d = waga wierzchołka stopnia d.
- $0 = w_0 = w_1 = w_2 < w_3 \leq w_4 \leq w_5 \leq w_6 \leq w_{7+} = 1$.
- Nie rozważamy juz $w_7, w_8, \ldots$ osobno bo nie tam będzie wąskie gardło.
- Dla wierzchołka stopnia d czynnik pracy $\lambda(w_d + d \min_{k \leq d}(w_k - w_{k-1}), w_d + d \cdot w_3)$.
- Albo dokładniej: dla wierzchołka v stopnia d, który...
  - ... ma samych sąsiadów stopnia 3, czynnik pracy $\lambda(w_d + d \cdot w_3, w_d + d \cdot w_3 + [\text{sąsiedzi sąsiadów}])$.
  - ... ma samych sąsiadów stopnia d, czynnik pracy $\lambda(w_d + d \cdot (w_d - w_{d-1}), w_d + d \cdot w_d + [\text{sąsiedzi sąsiadów}])$.
  - ... itd dla wszystkich możliwości sąsiedztwa v (jest ich kilka tysięcy).
- **Problem:** Jak optymalizować wagi gdy jest ich tak dużo?
  - local search (simulated annealing),
  - lub algorytm dokładny: Eppstein *Quasiconvex analysis of multivariate recurrence equations for backtracking algorithms*, TALG 2006

Najliczniejszy zbiór niezależny: mierz i zwyciężaj

Twierdzenie [Fomin, Grandoni, Kratsch 2006]

Ten sam algorytm (odrobinkę podkręcony) działa w czasie $O(2^{0.288n}) = O(1.221^n)$.

Mierz i zwyciężaj: Inne problemy, inne miary

Eppstein 2003: TSP w grafach stopnia co najwyżej 3, czas $O^*(2^{n/3})$:

- Rozwiązujemy ogólniejszy problem: dodatkowo dany zbiór krawędzi F, które **muszą** pojawić się w cyklu Hamiltona. W trakcie działania algorytmu nowe krawędzie są dodawane do F.

Mierz i zwyciężaj: Inne problemy, inne miary

Eppstein 2003: TSP w grafach stopnia co najwyżej 3, czas $O^*(2^{n/3})$:

- Rozwiązujemy ogólniejszy problem: dodatkowo dany zbiór krawędzi F, które **muszą** pojawić się w cyklu Hamiltona. W trakcie działania algorytmu nowe krawędzie są dodawane do F.
- Okazuje się, że gdy $G - F$ składa się wyłącznie z (rozłącznych) cykli długości 4 to problem można rozwiązać wielomianowo. Niech C będzie zbiorem takich cykli.

Mierz i zwyciężaj: Inne problemy, inne miary

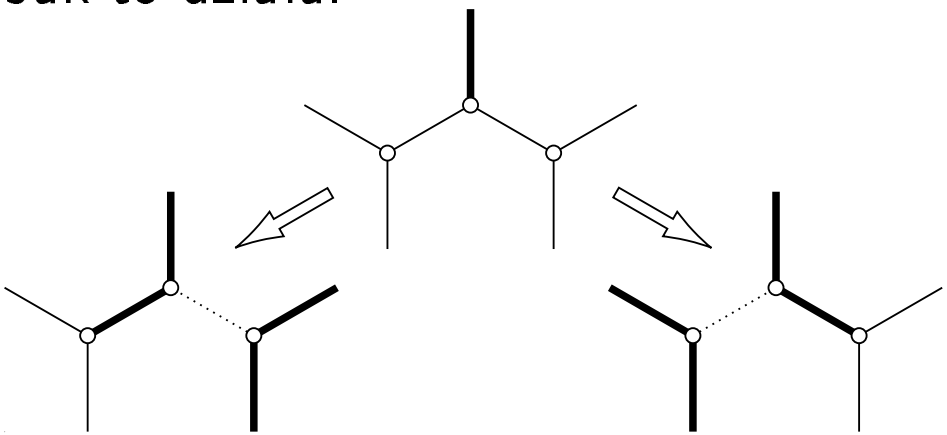
Eppstein 2003: TSP w grafach stopnia co najwyżej 3, czas $O^*(2^{n/3})$:

- Rozwiązujemy ogólniejszy problem: dodatkowo dany zbiór krawędzi F, które **muszą** pojawić się w cyklu Hamiltona. W trakcie działania algorytmu nowe krawędzie są dodawane do F.
- Okazuje się, że gdy $G - F$ składa się wyłącznie z (rozłącznych) cykli długości 4 to problem można rozwiązać wielomianowo. Niech C będzie zbiorem takich cykli.
- Miara: $\mu(G) = |V(G)| - |F| - |C|$.

Mierz i zwyciężaj: Inne problemy, inne miary

Eppstein 2003: TSP w grafach stopnia co najwyżej 3, czas $O^*(2^{n/3})$:

- Rozwiązujemy ogólniejszy problem: dodatkowo dany zbiór krawędzi F, które **muszą** pojawić się w cyklu Hamiltona. W trakcie działania algorytmu nowe krawędzie są dodawane do F.
- Okazuje się, że gdy $G - F$ składa się wyłącznie z (rozłącznych) cykli długości 4 to problem można rozwiązać wielomianowo. Niech C będzie zbiorem takich cykli.
- Miara: $\mu(G) = |V(G)| - |F| - |C|$.
- Jak to działa:



Zamiast czynnika pracy $\lambda(2,2) = 2^{1/2}$ mamy $(3,3) = 2^{1/3}$

Mierz i zwyciężaj: Inne problemy, inne miary

Eppstein 2005: $(4,2)$-CSP.
Problem podobny jak $(3,2)$-CSP, ale mamy 4 kolory zamiast 3.

- waga zmiennych, które mają ≤ 2 dozwolone kolory $w_{2-} = 0$.
- waga zmiennych, które mają 3 dozwolone kolory $w_3 = 1$.
- waga zmiennych, które mają 4 dozwolone kolory $w_4 = 2 - \varepsilon$, $\varepsilon = 0.0955$.
- Czas: $O(1.3645^{n_3+(2-\varepsilon)n_4})$, czyli $O(1.3645^n)$ dla $(3,2)$-CSP oraz 1.8702^n dla $(4,2)$-CSP.

Inne ciekawe wyniki korzystające z metody „mierz i zwyciężaj”

- Najmniejszy zbiór dominujący $O(2^{0.61n})$ [Fomin, Grandoni, Kratsch 05]
- Najmniejszy zbiór rozcyklający (FVS) $O(1.76^n)$ [Fomin, Gaspers, Pyatkin, Razgon 08]
- Bandwidth $O^*(4.83^n)$ [Cygan, Pilipczuk 08]

Technika: programowanie dynamiczne (po podzbiorach)

TSP (Problem komiwojażera)

Problem

Egzemplarz:
Graf pełny $G = (V = \{1, \ldots, n\}, E = \{xy \ : \ x, y \in V, x \neq y\})$,
funkcja $w : V^2 \to \mathbb{R}$.

Problem:
Znaleźć w G cykl Hamiltona o najmniejszej możliwej wadze (sumie wag krawędzi).

TSP (Problem komiwojażera: algorytm)

Algorytm [Held, Karp 1962]

Dla dowolnego $S \subseteq V$ oraz $v \in V$ niech $T[S, v]$ będzie najlżejszą ścieżką, która

- Zaczyna się w wierzchołku 1,
- odwiedza każdy wierzchołek z $S - \{v\}$,
- kończy się w v.

TSP (Problem komiwojażera: algorytm)

Algorytm [Held, Karp 1962]

Dla dowolnego $S \subseteq V$ oraz $v \in V$ niech $T[S, v]$ będzie najlżejszą ścieżką, która

- Zaczyna się w wierzchołku 1,
- odwiedza każdy wierzchołek z $S - \{v\}$,
- kończy się w v.

Wówczas:

- $T[\{v\}, v] = w(1, v)$,

TSP (Problem komiwojażera: algorytm)

Algorytm [Held, Karp 1962]

Dla dowolnego $S \subseteq V$ oraz $v \in V$ niech $T[S, v]$ będzie najlżejszą ścieżką, która

- Zaczyna się w wierzchołku 1,
- odwiedza każdy wierzchołek z $S - \{v\}$,
- kończy się w v.

Wówczas:

- $T[\{v\}, v] = w(1, v)$,
- $T[S, v] = \min_{x \in S - \{v\}}(T[S - \{v\}, x] + w(x, v))$.

TSP (Problem komiwojażera: algorytm)

Algorytm [Held, Karp 1962]

Dla dowolnego $S \subseteq V$ oraz $v \in V$ niech $T[S, v]$ będzie najlżejszą ścieżką, która

- Zaczyna się w wierzchołku 1,
- odwiedza każdy wierzchołek z $S - \{v\}$,
- kończy się w v.

Wówczas:

- $T[\{v\}, v] = w(1, v)$,
- $T[S, v] = \min_{x \in S - \{v\}}(T[S - \{v\}, x] + w(x, v))$.
- A więc tablicę T można wypełnić w czasie $O(n^2 2^n)$

TSP (Problem komiwojażera: algorytm)

Algorytm [Held, Karp 1962]

Dla dowolnego $S \subseteq V$ oraz $v \in V$ niech $T[S, v]$ będzie najlżejszą ścieżką, która

- Zaczyna się w wierzchołku 1,
- odwiedza każdy wierzchołek z $S - \{v\}$,
- kończy się w v.

Wówczas:

- $T[\{v\}, v] = w(1, v)$,
- $T[S, v] = \min_{x \in S - \{v\}}(T[S - \{v\}, x] + w(x, v))$.
- A więc tablicę T można wypełnić w czasie $O(n^2 2^n)$
- Następnie zwracamy $\min\{T[\{2, \ldots, n\}, j] + w(j, 1) \ : \ j \in \{2, \ldots, n\}\}$.

Wniosek

Problem TSP można rozwiązać w czasie $O(n^2 2^n)$ i pamięci $O(n2^n)$.

Liczba chromatyczna

k-kolorowanie

k-kolorowaniem grafu $G = (V, E)$ nazywamy dowolną funkcję $c : V \to \{1, \ldots, k\}$ taką, że dla dowolnej krawędzi $xy \in E$, $c(x) \neq c(y)$.

Liczba chromatyczna

Liczbą chromatyczną grafu G nazywamy najmniejsze k takie, że istnieje k-kolorowanie G i oznaczamy $\chi(G)$.

Problem

Egzemplarz:
Graf nieskierowany $G = (V, E)$.

Problem:
Znaleźć $\chi(G)$.

Liczba chromatyczna: algorytm Lawlera

Obserwacja

Dla dowolnego grafu G istnieje $\chi(G)$-kolorowanie G, w którym wierzchołki pomalowane pewnym kolorem tworzą maksymalny zbiór niezależny.

Algorytm [Lawler 1976]

1: $\chi[\emptyset] \leftarrow 0$.
2: **for each** $S \subseteq V$ w porządku rosnącej mocy, **do**
3: $\quad \chi[S] \leftarrow |S|$.
4: $\quad$ **for each** $I \subseteq S$ wygenerowanego przez algorytm #MIS$(G[S])$ **do**
5: $\quad\quad \chi[S] \leftarrow \min\{\chi[S], 1 + \chi[S - I]\}$.
6: $\quad$ *niezmiennik:* $\chi[S] = \chi(G[S])$.

Liczba chromatyczna: algorytm Lawlera — analiza

Ze wzoru dwumianowego Newtona:

$$\sum_{k=0}^{n} \binom{n}{k} 3^{k/3} = (1 + 3^{1/3})^n < 2.4422^n.$$

Złożoność czasowa

Algorytm Lawlera znajduje liczbę chromatyczną w czasie $O(2.4422^n)$ i pamięci $O(2^n)$. Właściwie, znajduje on liczby chromatyczne wszystkich podgrafów indukowanych danego grafu.

3-kolorowanie: algorytm Lawlera

Obserwacja

Można sprawdzić w czasie wielomianowym, czy dany graf jest 2-kolorowalny (DFS).

Algorytm [Lawler 1976]

1: **for each** $I \subseteq V$ wygenerowanego przez algorytm #MIS($G[S]$) **do**
2: **if** $\chi([G[V - I]) \leq 2$ **then**
3: **return** "$\chi(G) \leq 3$"
4: **return** "$\chi(G) > 3$".

Wniosek

Powyższy algorytm sprawdza, czy dany graf jest 3-kolorowalny w czasie $O^*(3^{n/3}) = O(1.4423^n)$ i pamięci wielomianowej. (Łatwo go przerobić, żeby zwracał 3-kolorowanie.)

Programowanie dynamiczne po dekompozycji drzewowej

Przypomnienie

Wiele problemów grafowych można rozwiązać w czasie $O^*(c^{\mathrm{tw}(G)})$, gdzie c jest stałą, a $\mathrm{tw}(G)$ oznacza szerokość drzewową wejściowego grafu G. (Patrz wykład Dr. Daniela Marxa).
Np. najliczniejszy zbiór niezależny można znaleźć w czasie $O(2^{\mathrm{tw}(G)}n)$.

Twierdzenie [Lipton, Tarjan]

Jeśli G jest grafem planarnym o n wierzchołkach, to $\mathrm{tw}(G) = O(\sqrt{n})$.

Twierdzenie [Fomin, Høie]

Dla dowolnego $\epsilon > 0$, jeśli G jest grafem subkubicznym (o maksymalnym stopniu wierzchołka 3) i o $n > f(\epsilon)$ wierzchołkach, to $\mathrm{tw}(G) \leq \frac{n}{6} + \epsilon$.

Programowanie dynamiczne po dekompozycji drzewowej

Wniosek

Najliczniejszy zbiór niezależny można znaleźć w czasie i pamięci:

- $2^{O(\sqrt{n})}$ dla grafów planarnych,
- $O^*(2^{n/6})$ dla grafów subkubicznych.

Podobnie, dla grafów subkubicznych możemy dostać algorytmy:

- $O^*(2^{n/6})$ dla problemu MAX-CUT.
- $O^*(3^{n/6})$ dla problemu minimalnego zbioru dominującego.

Technika: Dziel (na wykładniczą liczbę egzemplarzy) i zwyciężaj

TSP w pamięci wielomianowej [Bjorklund, Husfeldt 2005]

Poniższy algorytm znajduje najlżejszą ścieżkę prostą od s do t spośród ścieżek które odwiedzają wszystkie wierzchołki A i żadnych innych. (Jak zwykle, dla uproszczenia znajdujemy tylko wagę najlżejszej ścieżki).

TSP(s, t, A)

1. Jeśli $s = t$ zwróć 0, jeśli $s \neq t$ i $A = \emptyset$, zwróć $w(s, t)$.
2. Wygeneruj wszystkie podziały A na trzy rozłączne częsci: $V = L \cup \{x\} \cup R$, takie że $|L| = \lfloor |A - 1|/2 \rfloor$.

TSP w pamięci wielomianowej [Bjorklund, Husfeldt 2005]

Poniższy algorytm znajduje najlżejszą ścieżkę prostą od s do t spośród ścieżek które odwiedzają wszystkie wierzchołki A i żadnych innych. (Jak zwykle, dla uproszczenia znajdujemy tylko wagę najlżejszej ścieżki).

TSP(s, t, A)

1. Jeśli $s = t$ zwróć 0, jeśli $s \neq t$ i $A = \emptyset$, zwróć $w(s, t)$.
2. Wygeneruj wszystkie podziały A na trzy rozłączne częsci: $V = L \cup \{x\} \cup R$, takie że $|L| = \lfloor |A - 1|/2 \rfloor$.
3. Dla każdego podziału oblicz TSP(s, x, L) + TSP(x, t, R).
4. Zwróć najmniejszy z obliczonych wyników.

TSP w pamięci wielomianowej [Bjorklund, Husfeldt 2005]

Poniższy algorytm znajduje najlżejszą ścieżkę prostą od s do t spośród ścieżek które odwiedzają wszystkie wierzchołki A i żadnych innych. (Jak zwykle, dla uproszczenia znajdujemy tylko wagę najlżejszej ścieżki).

TSP(s, t, A)

1. Jeśli $s = t$ zwróć 0, jeśli $s \neq t$ i $A = \emptyset$, zwróć $w(s, t)$.
2. Wygeneruj wszystkie podziały A na trzy rozłączne częsci: $V = L \cup \{x\} \cup R$, takie że $|L| = \lfloor |A - 1|/2 \rfloor$.
3. Dla każdego podziału oblicz TSP(s, x, L) + TSP(x, t, R).
4. Zwróć najmniejszy z obliczonych wyników.

$$T(n) = n\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor} \cdot 2 \cdot T\left(\left\lceil \frac{n-1}{2} \right\rceil\right)$$

TSP w pamięci wielomianowej [Bjorklund, Husfeldt 2005]

Poniższy algorytm znajduje najlżejszą ścieżkę prostą od s do t spośród ścieżek które odwiedzają wszystkie wierzchołki A i żadnych innych. (Jak zwykle, dla uproszczenia znajdujemy tylko wagę najlżejszej ścieżki).

TSP(s, t, A)

1. Jeśli $s = t$ zwróć 0, jeśli $s \neq t$ i $A = \emptyset$, zwróć $w(s, t)$.
2. Wygeneruj wszystkie podziały A na trzy rozłączne częsci: $V = L \cup \{x\} \cup R$, takie że $|L| = \lfloor |A - 1|/2 \rfloor$.
3. Dla każdego podziału oblicz TSP(s, x, L) + TSP(x, t, R).
4. Zwróć najmniejszy z obliczonych wyników.

$$T(n) = n\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor} \cdot 2 \cdot T\left(\left\lceil \frac{n-1}{2} \right\rceil\right)$$

$$T(n) \leq 2^n \cdot 2 \cdot T\left(\left\lceil \frac{n-1}{2} \right\rceil\right)$$

TSP w pamięci wielomianowej [Bjorklund, Husfeldt 2005]

Poniższy algorytm znajduje najlżejszą ścieżkę prostą od s do t spośród ścieżek które odwiedzają wszystkie wierzchołki A i żadnych innych. (Jak zwykle, dla uproszczenia znajdujemy tylko wagę najlżejszej ścieżki).

TSP(s, t, A)

1. Jeśli $s = t$ zwróć 0, jeśli $s \neq t$ i $A = \emptyset$, zwróć $w(s, t)$.
2. Wygeneruj wszystkie podziały A na trzy rozłączne częsci: $V = L \cup \{x\} \cup R$, takie że $|L| = \lfloor |A - 1|/2 \rfloor$.
3. Dla każdego podziału oblicz TSP(s, x, L) + TSP(x, t, R).
4. Zwróć najmniejszy z obliczonych wyników.

$$T(n) = n\binom{n-1}{\lfloor \frac{n-1}{2} \rfloor} \cdot 2 \cdot T\left(\left\lceil \frac{n-1}{2} \right\rceil\right)$$

$$T(n) \leq 2^n \cdot 2 \cdot T\left(\left\lceil \frac{n-1}{2} \right\rceil\right) \leq 2^n \cdot 2 \cdot 2^{n/2} \cdot 2 \cdot 2^{n/4} \cdot 2 \cdots = 2^{2n} \cdot 2^{\log n} = O^*(4^n)$$

TSP w pamięci wielomianowej [Bjorklund, Husfeldt 2005]

Waga najlżejszego cyklu Hamiltona jest równa

$$\min_{j=2,\ldots,n} \mathrm{TSP}(1, j, V \setminus \{1, j\}) + w(j, 1).$$

Wniosek

TSP można rozwiązać w czasie $O^*(4^n)$ i pamięci wielomianowej.

Uwaga

Nikt nie umie lepiej, chyba że wagi są ograniczone (np. przez wielomian).

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

Obserwacja

Rozważmy dowolne k-kolorowanie grafu $G = (V, E)$. Wtedy:

1. $V = I_1 \cup I_2 \cup \cdots \cup I_k$, gdzie I_j są zbiorami niezależnymi.

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

Obserwacja

Rozważmy dowolne k-kolorowanie grafu $G = (V, E)$. Wtedy:

1. $V = I_1 \cup I_2 \cup \cdots \cup I_k$, gdzie I_j są zbiorami niezależnymi.
2. Bez straty ogólności największy zbiór I_{j^*} jest maksymalnym zbiorem niezależnym.

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

Obserwacja

Rozważmy dowolne k-kolorowanie grafu $G = (V, E)$. Wtedy:

1. $V = I_1 \cup I_2 \cup \cdots \cup I_k$, gdzie I_j są zbiorami niezależnymi.
2. Bez straty ogólności największy zbiór I_{j^*} jest maksymalnym zbiorem niezależnym.
3. Pozostałe zbiory można podzielić na 2 rodziny, z których każda pokrywa $< |V|/2$ wierzchołków.

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

Obserwacja

Rozważmy dowolne k-kolorowanie grafu $G = (V, E)$. Wtedy:

1. $V = I_1 \cup I_2 \cup \cdots \cup I_k$, gdzie I_j są zbiorami niezależnymi.
2. Bez straty ogólności największy zbiór I_{j^*} jest maksymalnym zbiorem niezależnym.
3. Pozostałe zbiory można podzielić na 2 rodziny, z których każda pokrywa $< |V|/2$ wierzchołków.
   (Dokładamy zbiory aż suma ich elementów przekroczy $|V|/2$. Wtedy wyjmujemy ostatni zbiór I_ℓ. Ponieważ $|I_\ell| < |I_{j^*}|$ to suma elementów pozostałych zbiorów jest mniejsza niż $|V|/2$.)

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

Obserwacja

Rozważmy dowolne k-kolorowanie grafu $G = (V, E)$. Wtedy:

1. $V = I_1 \cup I_2 \cup \cdots \cup I_k$, gdzie I_j są zbiorami niezależnymi.
2. Bez straty ogólności największy zbiór I_{j^*} jest maksymalnym zbiorem niezależnym.
3. Pozostałe zbiory można podzielić na 2 rodziny, z których każda pokrywa $< |V|/2$ wierzchołków.

Algorytm

1. Dla każdego z $3^{n/3}$ kandydatów I na zbiór I_{j^*} ...
2. Dla każdego podziału $V - I$ na części A, B rozmiaru $< |V|/2$ każda,
3. Rekurencyjnie oblicz $\chi(G[A])$, $\chi(G[B])$

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

Obserwacja

Rozważmy dowolne k-kolorowanie grafu $G = (V, E)$. Wtedy:

1. $V = I_1 \cup I_2 \cup \cdots \cup I_k$, gdzie I_j są zbiorami niezależnymi.
2. Bez straty ogólności największy zbiór I_{j^*} jest maksymalnym zbiorem niezależnym.
3. Pozostałe zbiory można podzielić na 2 rodziny, z których każda pokrywa $< |V|/2$ wierzchołków.

Algorytm

1. Dla każdego z $3^{n/3}$ kandydatów I na zbiór I_{j^*} ...
2. Dla każdego podziału $V - I$ na części A, B rozmiaru $< |V|/2$ każda,
3. Rekurencyjnie oblicz $\chi(G[A])$, $\chi(G[B])$
4. zwróć najmniejszą spośród sum $1 + \chi(G[A]) + \chi(G[B])$.

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

$$T(n) \leq 3^{n/3} \cdot 2^n \cdot T(n/2) = O((2 \cdot 3^{1/3})^{2n}) = O(8.33^n)$$

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

$$T(n) \leq 3^{n/3} \cdot 2^n \cdot T(n/2) = O((2 \cdot 3^{1/3})^{2n}) = O(8.33^n)$$

Wniosek

Liczbę chromatyczną można znaleźć w czasie $O(8.33^n)$ i pamięci wielomianowej.

Liczba chromatyczna w pamięci wielomianowej [B,H 2005]

$$T(n) \leq 3^{n/3} \cdot 2^n \cdot T(n/2) = O((2 \cdot 3^{1/3})^{2n}) = O(8.33^n)$$

Wniosek

Liczbę chromatyczną można znaleźć w czasie $O(8.33^n)$ i pamięci wielomianowej.

Uwaga

Można lepiej, dużo lepiej: $O(2.24^n)$. Jak? Dowiemy się w maju.