Zastosowanie łańcuchów ograniczających do znajdowania poprawnego kolorowania grafu

Antoni Szustakowski

Wstęp

Łańcuchy ograniczające to technika ułatwiająca zastosowanie Łańcuchów Markowa. W swojej pracy [1] Mark Huber wymienia: teoretyczne ograniczenie na czas mieszania (pod pewnymi warunkami), empiryczne ograniczenie na czas mieszania (których dokładność można dowieść) oraz konstrukcję algorytmów pozwalających na dokładne próbkowanie. Wymienione jest również wiele zastosowań, między innymi kolorowanie grafu, czy model antyferromagnetyczny Pottsa. W tym projekcie zająłem się problemem kolorowania grafu.

Wprowadzenie formalne

Dane są 2 zbiory: C i V. Zbiór C odpowiada możliwym wartościom, natomiast zbiór V - wymiarom. Bazową przestrzenią stanóW dla X będzie $\Omega \subseteq C^V$. Natomiast dla Y - $(2^C)^V$. Istnieją dwie definicje (formy) łańcuchów ograniczających, natomiast na potrzeby tego projektu skorzystam z tak zwanej Definicji 1:

Definicja:
Powiemy, że Y jest *łańcuchem ograniczającym dla X*, jeśli:

$$X_t(v) \in Y_t(v) \forall v \Rightarrow X_{t+1}(v) \in Y_{t+1}(v) \forall v$$

Kolorownanie grafu

Definicja:
Dany jest graf G=(V,E) (V - zbiór wierzchołków, E - zbiór krawędzi). Niech n oznacza liczbę wierzchołków. *Poprawnym k-kolorowaniem grafu G* nazwiemy:

$$x : V \to \{1, ..., k\}^n, \forall e \in E, \forall v, w \in e : x(v) \neq x(w)$$

Innymi słowy, jeśli dwa wierzchołki są połączone krawędzią, nie mogą być pokolorowane na ten sam kolor. Nie dla każdej liczby kolorów k istnieje poprawne k-kolorowanie. Między innymi, dla grafów pełnych minimalną liczbą kolorów (liczbą chromatyczną), które należy użyć jest liczba wierzchołków. Poniższe twierdzenie ułatwi implementację algorytmu:

Twierdzenie (ograniczenie górne na liczbę chromatyczną):
G - dowolny graf, $\chi(G)$ - liczba chromatyczna grafu G, $\Delta(G)$ - maksymalny stopień grafu G, wtedy:

$$\chi(G) \leq \Delta(G) + 1$$

Cel

Celem tego projektu jest próbkowanie z rozkładu jednostajnego poprawnych kolorowań grafu, przy zadanej liczbie kolorów. Próbkowanie to będzie oparte na algorytmie Proppa-Wilsona. Rozkład jednostajny będzie uzyskiwany dzięki poniższemu twierdzeniu:

Twierdzenie (o wyniku algorytmu Proppa-Wilsona):
Niech P będzie macierzą przejścia nieprzywiedlnego i nieokresowego łańcucha Markowa z przestrzenią

stanów $S = \{s_1, s_2, ..., s_n\}$ i rozkładem stacjonarnym $\pi = (\pi_1, ..., \pi_n)$. Niech Φ będzie przekształceniem dla macierzy P z algorytmu Proppa-Wilsona. Rozważmy algorytm Proppa-Wilsona z $(N_1, N_2, ...) = (1, 2, 4, 8, ...)$. Przypuśćmy, że algorytm zbiega z prawdopodobieństwem 1 i niech Y będzie wynikiem tego algorytmu. Wtedy dla każdego $i \in 1, ..., n$ zachodzi:

$$P(Y = s_i) = \pi_i$$

Algorytm Coupling From the Past (CFTP)

Użycie Próbnika Gibbsa

Głównym komponentem algorytmu CFTP będzie Próbnik Gibbsa. Zgodnie z powyższym celem, chcielibyśmy próbkować poprawne kolorowania $X \subset C^V$ według rozkładu jednostajnego π. Przypuśćmy, że możemy bardzo łatwo uzyskać prawdopodobieństwa warunkowe $\pi(X(v) = i | X(V \setminus \{v\}) = \xi)$ dla wszystkich kombinacji $v \in V$, $i \in C$ oraz $\xi \in C^{V \setminus \{v\}}$. W przypadku praktycznym funkcja akceptacji będzie pełniła rolę prawdopodobieństw warunkowych. W Próbniku Gibbsa w iteracji $t + 1$ będziemy jednostajnie losować wierzchołek $v \in V$, który będzie aktualizowany zgodnie z funkcją akceptacji. Pozostałe wierzchołki nie zmienią swoich kolorów:

$$X_{t+1}(w) = X_t(w), \forall w \in V \setminus \{v\}$$

$$X_{t+1}(v) \stackrel{d}{=} \pi(\cdot | X_t(V \setminus \{v\}))$$

W oczywisty sposób możemy taką procedurą uzyskać każdy z możliwych stanóW w C^V, więc łańcuch $\{X_t\}_{t \in (N)}$ jest nieprzywiedlny. Dodatkowo funkcja akceptacji z niezerowym prawdopodobieństwem może nakazać łańcuchowi pozostanie w danym stanie, zatem $\{X_t\}$ jest nieokresowy. Ostatecznie $\{X_t\}$ jest ergodyczny (bo operujemy na skończonej przestrzeni stanów), zatem ma rozkład stacjonarny π, z uwagi na dobrane prawdopodobieństwa przejścia i zbiega do niego przy $t \to \infty$. Zatem warunek z twierdzenia o wyniku algorytmu Proppa-Wilsona jest spełniony i jesteśmy w stanie próbkować z rozkładu π.

Zastosowanie łańcuchów ograniczających

Problem z CFTP

Algorytm CFTP startuje $|C^V|$ łańcuchów w pewnym czasie -t, następnie wykorzystując ciąg zmiennych jednostajnych, aktualizuje tę samą współrzędną w konkretnej iteracji dla każdego z wystartowanych łańcuchów. Jeśli po t krokach (czyli w czasie 0) wszystkie łańcuchy znajdą się w tym samym stanie, algorytm się kończy. Natomiast w przeciwnym wypadku, liczba iteracji jest zwiększana dwukrotnie (do 2t). Jaki jest koszt takiego działania? Przede wszystkim, dla dużych przestrzeni stanów (jakimi są między innymi wszystkie możliwe kolorowania grafów) koszt przechowywania informacji o danej iteracji jest ogromny. Dodatkowo, sama złożoność algorytmu jest ogromna. Można znacznie uprościć zadanie komputerowi, dzięki zastosowaniu łańcuchów ograniczających.

Inny sposób na zapisanie warunku stopu algorytmu CFTP

W algorytmie CFTP warunkiem stopu jest zatrzymanie się wszystkich łańcuchów w danym stanie. CFTP wykorzystuje przekształcenie losowe $\Phi : C^V \times [0, 1] \to C^V$, zatem, aby warunek stopu był spełniony wystarczy, by od pewnego miejsca w czasie -t:

$$\psi(X_{-t}(s), U_{-t}) = const., \forall s \in C^V$$

Czyli wystarczy, aby przekształcenie losowe od pewnego miejsca było stałe.

Pomysł

Idea na użycie łańcuchów ograniczających zaprezentowana przez Marka Hubera polega na uruchomieniu dwóch łańcuchów Markowa (zgodnie z procedurą Proppa-Wilsona): X - odpowiadającego za pokolorowanie każdego z wierzchołków, Y - odpowiadającego za wszystkie możliwe pokolorowania każdego z wierzchołków w danej iteracji. Główne operacje będą wykonywane na łańcuchu Y: będziemy sprawdzać wszystkie możliwe

kolorowania dla losowo wybranego wierzchołka i jeden z wybranych kolorów dla Y będzie pokolorowaniem wierzchołka odpowiadającego w X. Procedurę będziemy powtarzać dopóki dla wszystkich wierzchołków liczba możliwych pokolorowań będzie 1, co w oczywisty sposób oznacza, że dany wierzchołek został pokolorowany i nie ma potrzeby aktualizacji jego koloru.

Startowanie łańcucha

Chcemy spełnić definicję 1. łańcucha ograniczającego w każdym obrocie pętli. W swojej pracy Huber podaje, że najprostszą drogą do osiągnięcia tego jest zadanie: $Y_0(v) = C, \forall v \in V$. Wtedy mamy pewność (dzięki konstrukcji algorytmu), że łańcuch Y zawsze będzie ograniczał łańcuch X. Natomiast łańcuch X wystartujmy od stanu, w którym każdy wierzchołek będzie miał taki sam kolor, tzn. $X_0(v) = 1, \forall v \in V$.

Algorytm

Niech n oznacza liczbę wierzchołków w grafie G=(V,E), Δ - maksymalny stopień G, $t \in \mathbb{N}$ - aktualną iterację, it - maksymalną liczbę iteracji, a k - maksymalną liczbę kolorów. Głównym algorytmem dla zadanego problemu będzie:

1. **Dla t=1,2,...,it**:
2. **Jeżeli** $\forall v \in V : |Y_{t-1}(v)| = 1$:
   - Przerwij i zwróć $X_{t-1}(v)$.
   - W przeciwnym wypadku przejdź do punktu 3.
3. **Niech** $Y_t = Y_{t-1}$ oraz $X_t = X_{t-1}$.
4. **Wylosuj jednostajnie** $v \in [n]$, niech N_v oznacza zbiór sąsiadów v.
5. **Jeżeli** $|Y_t(v)| = 1$:
   - Wróć do punktu 3.
   - W przeciwnym przypadku przejdź do punktu 5.
6. **Niech** $Y_t(v) \leftarrow \varnothing$.
7. **Dopóki** $c \notin \bigcup_{w \in Nv} Y_t(w)$ lub $|Y_t(v)| > \Delta$:
8. **Wylosuj jednostajnie** $c \in [k]$
9. **Jeżeli** $\forall w \in Nv : Y_{t+1}(w) \neq \{c\}$:
   - $Y_t(v) \leftarrow Y_t(v) \cup \{c\}$
10. **Niech** $X_t(v) = c$
11. **Jeżeli** warunek z punktu 2. dla iteracji o numerze it nie jest spełniony:
    - it $\leftarrow 2 * it$
    - **Powtórz** algorytm od samego początku

Logika algorytmu

W każdym kolejnym obrocie pętli najpierw sprawdzamy, czy każdy wierzchołek ma przyporządkowany już kolor. Jeśli tak - to nie ma sensu dalej szukać kolejnego kolorowania. Jeśli nie - wybieramy losowo wierzchołek, który nie ma jeszcze przyporządkowanego koloru. Jego możliwe kolorowania (zapisane w łańcuchu Y) stają się zbiorem pustym, bo chcemy dodać nowe możliwe kolory dla danego wierzchołka. W szczególności (dzięki losowemu wyborowi kolorów) ta operacja pozwala na dodanie tylko jednego koloru do łańcucha Y. Dodajemy losowo wybrany kolor, o ile żaden z sąsiadów nie ma wybranego tego koloru (innymi słowy nie ma tylko jednej możliwości na pokolorowanie i ta możliwość to wylosowany kolor). Powtarzamy operację losowania aż znajdziemy kolor, którego nie ma żaden z sąsiadów (w łańcuchu Y) lub przekroczymy maksymalny stopień

grafu, bo - zgodnie z twierdzeniem - wtedy mamy pewność, że pewien z wybranych kolorów będzie właściwym. Wierzchołek kolorujemy na ostatni z wylosowanych kolorów, który spełnił nasze warunki. Powyższą operację powtarzamy zadaną liczbę iteracji.

Uzasadnienie formalne

W swojej pracy [2] Haggstrom i Nelander podają twierdzenie wraz z dowodem, kiedy powyższy algorytm będzie zbiegał oraz co będzie zwracał.

Dla dowolnego $a \in (2^C)^V$ oznaczmy $card(a)$ jako liczbę elementów $s \in C$, takich, że $s \in a$.
Twierdzenie: Jeżeli istnieje $n < \infty$, takie, że:

$$P(\forall v \in V : card(Y_n(v)) = 1 | \forall v \in V : Y_0(v) = C) > 0$$

To powyższy algorym P-prawie na pewno zbiega i zwraca nieobciążoną próbkę z rozkładu π.

Dobieranie liczby kolorów

Dobranie odpowiedniej liczby kolorów jest kluczowe, aby łańcuchy przechodziły z jednego stanu do istotnie różnego kolejnego stanu. Jeśli dobierzemy ich zbyt mało, kolorowanie będzie niemożliwe lub algorytm będzie wyłącznie losowo dobierał kolory. Natomiast jeśli dobierzemy ich zbyt dużo - z dużym prawdopodobieństwem każdy wierzchołek zostanie pokolorowany na inny kolor.

W swojej pracy Huber podaje 2 liczby:

1. $k > \frac{11\Delta}{6}$ - jako najmniejszą liczbę pozwalającą na przemieszanie się algorytmu w czasie wielomianowym - podaną przez Vigodę.
2. $k \geq \Delta(\Delta + 2)$ - jako liczbę pozwalającą na ustalenie ograniczenia dolnego na prawdopodobieństwo znalezienia odpowiedniego kolorowania w zadanej liczbie kroków.

Twierdzenie:
Niech:

$$\beta = 1 - \frac{\frac{1-(\Delta+1)\Delta}{k-\Delta+1}}{n}$$

Jeśli liczba kolorów jest minimum $\Delta(\Delta+2)$ to prawdopodobieństwo, że łańcuch znajdzie poprawne kolorowanie w $log_\beta n + \theta$ kroków wynosi co najmniej $1 - \beta^\theta$.

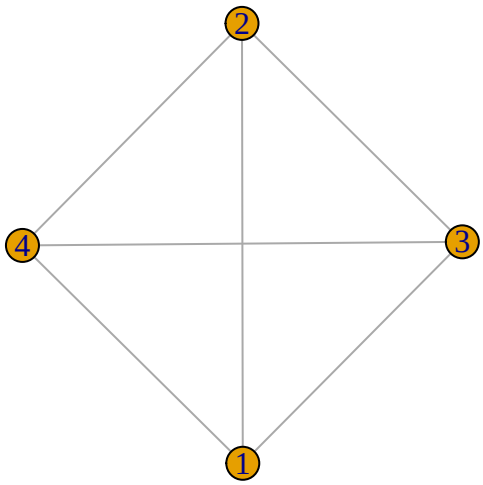
Zastosowane liczby kolorów

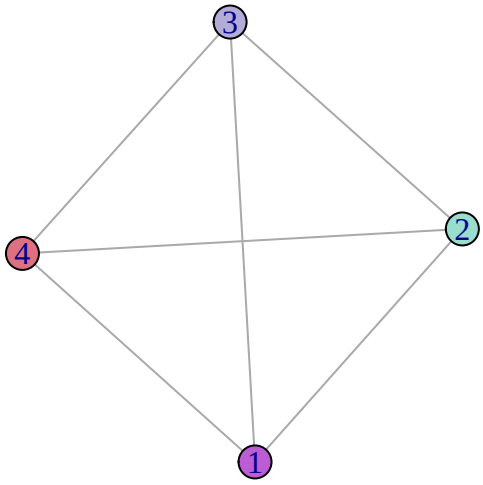
W tym projekcie dla małych grafów (o liczbie wierzchołków do 9-10) zastosowałem liczbę $k = \lceil \frac{11\Delta}{6} \rceil$. Natomiast dla dużych grafów (o liczbie wierzchołków większej od 10) liczbę $k = \Delta(\Delta + 2)$. Taki wybór wynika jedynie z obserwacji działania funkcji - nie ma ścisłego dowodu na lepsze działanie każdej z tych liczb w określonych przypadkach.

Działanie algorytmu na konkretnych grafach

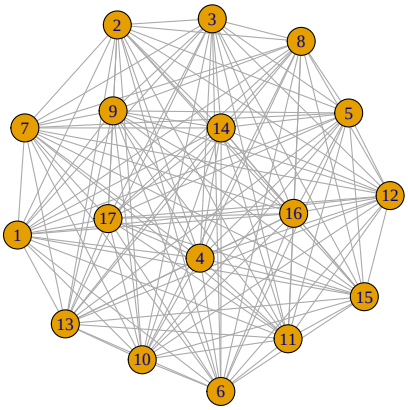
Zobaczmy, jak działa algorytm dla konkretnych grafów:

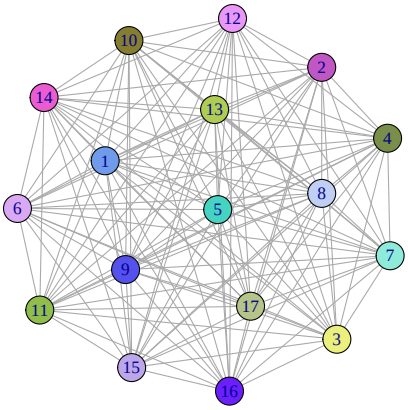
Niewielki graf pełny





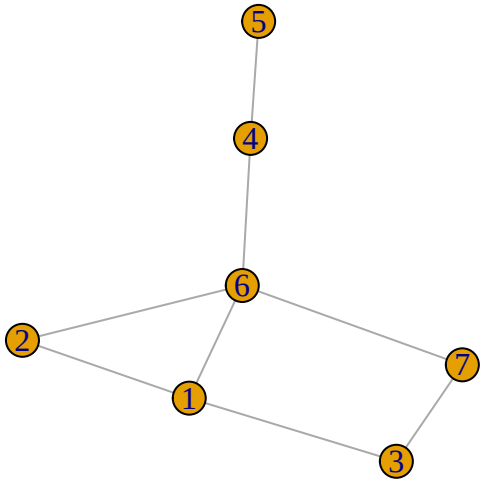
Duży graf pełny

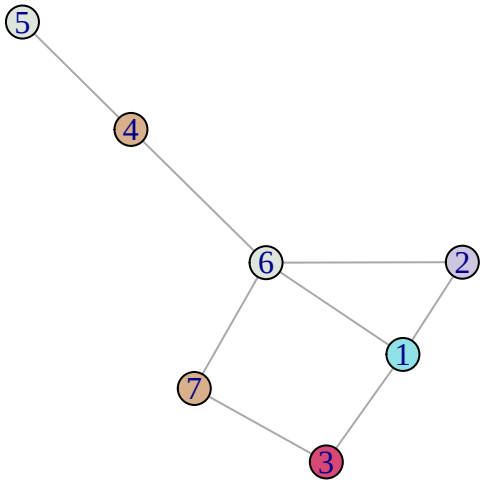




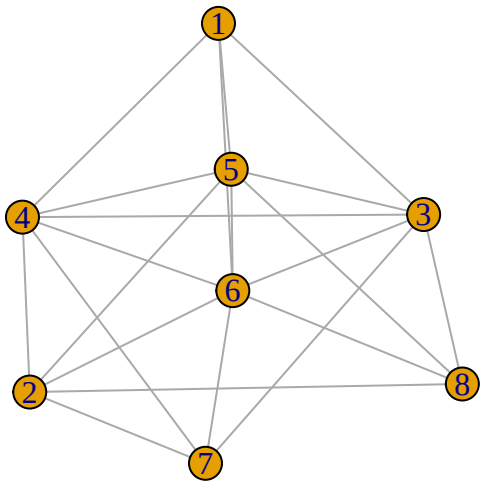

```
## [1] "Liczba różnych kolorów:"
## [1] 17
```

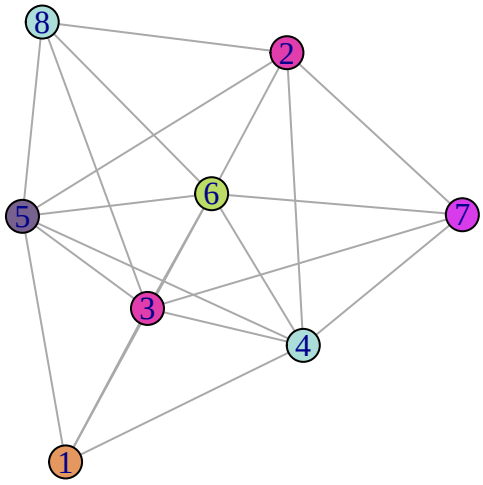
Graf z niewielką liczbą krawędzi





Graf z dużą liczbą krawędzi





Uzyskanie rozkładu stacjonarnego

Rozkład stacjonarny poprawnych kolorowań dla konkretnego grafu możemy uzyskać poprzez wykonanie dużą liczbę razy algorytmu i zliczenie częstości występowania każdego ze zwracanych stanów.

Podsumowanie

Powyżej zaprezentowany algorytm, bazujący na łańcuchach ograniczających, zwraca poprawne kolorowanie dla różnych typów grafów i dzięki niemu możemy uzyskać rozkład stacjonarny poprawnych kolorowań. Wyniki algorytmu empirycznie pokazują poprawność wcześniejszych rozważań teoretycznych. Zatem wykorzystanie łańcuchów ograniczających znacząco ułatwia pracę przy znajdowaniu rozkładu stacjonarnego i jest przy tym bardzo intuicyjne.