Algorytmy Zaawansowane - kolorowanie $Delta(G)$

Albert Sadowski, Piotr Stanek
Wydział Matematyki i Nauk Informacyjnych
Politechnika Warszawska

18 kwietnia 2013

1 Opis problemu

Kolorowanie wierzchołków grafu polega na przypisaniu wierzchołkom grafu kolorów, w taki sposób aby żadne dwa sąsiednie wierzchołki nie miały tego samego koloru. Mówimy, że kolorowanie jest poprawne, gdy końcom żadnej krawędzi nie przypisano tego samego koloru.

Liczbą chromatyczną grafu G nazywamy liczbę $\chi(G)$ równą najmniejszej możliwej liczbie kolorów potrzebnych do poprawnego pokolorowania wierzchołków grafu G.

Przypisując różne kolory to różnych wierzchołków grafu, zawsze otrzymamy poprawne kolorowanie:

$$1 \leqslant \chi(G) \leqslant n \tag{1}$$

gdzie n jest liczbą wierzchołków w grafie.

Twierdzenie *Vizinga* mówi, że dla grafu $G = (V, E)$, w którym największy stopień wierzchołka wynosi d. Wówczas krawędzie grafu G można pokolorować przy użyciu $d + 1$ kolorów; tj.:

$$d \leqslant \chi(G) \leqslant d + 1 \tag{2}$$

Twierdzenie *Brooksa* mówi, że dla grafu $G = (V, E)$, który jest spójnym grafem o największym stopniu równym d. Jeżeli G jest grafem pełnym lub składa się z pojedynczego cyklu o nieparzystej liczbie krawędzi, to $\chi(G) = d + 1$. We wszystkich pozostałych przypadkach $\chi(G) \leqslant d$.

Naszym zadaniem jest zaprojektowanie algorytmu kolorującego wierzchołki grafu (który nie jest grafem pełnym oraz który nie składa się z pojedynczego cyklu o nieparzystej liczbie krawędzi) spełniając: $\chi(g) \leqslant d$. Algorytm powiniem spełniać założenie o wielomianowej złożoności obliczeniowej.

Jedną z metod rozwiązania problemu kolorowania grafu, jest zastosowanie algorytmu zachłannego. Algorytmy zachłanne kolorując graf przyjmują wierzchołki 'jeden po drugim' przydzielając najmniejszy możliwy kolor. Poprzez odpowiednie ustalenie kolejności odwiedzania wierzchołków przez algorytm można otrzymać różne wyniki.

2 Algorytm

Bazując na metodzie opisanej przez *Bryant V.* w książce *Aspekty kombinatoryki*, proponujemy następujący algorytm.

Rozważając spójny graf $G = (V, E)$ składający się z n wierzchołków ($n \geqslant 3$), który nie jest pełny i nie składa się z pojedynczego cyklu o nieparzystej liczbie krawędzi:

1. W grafie G znajdź wierzołki rozspajające. Po usunięciu tych wierzchołków graf nie jest spójny. Dla każdego spójnego podgrafu otrzymanego z usunięcia wierzchołków rozspajających graf G:
   (a) Znajdź takie trzy wierzchołki u, v, w, że $uw, vw \in E, uv \notin E$ oraz usunięcie u oraz v (wraz ze wszystkimi dochodzącymi do nich krawędziami) pozostawi spójny graf G'.
   (b) Dla każdego wierzchołka x grafu G' wyznacz najkrótszą ścieżkę od w do x w grafie G'. Przypisz każdemu wierzchołkowi x odległość.
   (c) Przyporządkuj wierzchołkom grafu G etykiety w następujący sposób:
      i. Niech $v_1 = u, v_2 = v$.

ii. Niech v_3 będzie wierzchołkiem o najwyższej wartości przypisanej w pkt. 2.

iii. Pozostałe wierzchołki oznacz jako $v_4, v_5 \dots v_{n-1}$ tak, aby przypisane im wartości były rosnąco bliższe wierzchołkowi w (tj. tak że w grafie G' odległość np. z v_5 do w jest mniejsza bądź równa odległości z v_4 do w itd.).

iv. Niech $v_n = w$.

(d) Pokoloruj algorytmem zachłannym wierzchołki grafu G w kolejności $v_1, v_2, ..., v_n$.

2. Dla pierwszego podgrafu ze zbioru pokolorowanych podgrafów dołącz sąsiadujący wierzchołek rozspójniający, pokoloruj go.

3. Dołączaj kolejne sąsiadujące podgrafy. Jeżeli bieżące kolorowanie podgrafu nie pozwala dołączyć kolejnego, zmień kolorowanie w sposób umożliwiający ich połączenie.

Przypisując wierzchołkom porządek opisany w pkt. 3., mamy pewność, że każdy wierzchołek $v_i (i < 11)$ jest połączony z wierzchołkiem o większym wskaźniku (czyli z niepokolorowanym). Dla każdego wierzchołka będzie dostępny jeden z kolorów (nie przekraczający $Delta(G)$. W chwili kolorowania ostatniego wierzchołka v_n, jest on połączony z dwoma wierzchołkami (v_1 oraz v_2) o kolorze c_1, a więc pozostanie dla niego wolny kolor.

3 Analiza złożoności

1. **Procedura 1:**

   Znajdowanie wierzchołków rozspajających. Przeszukanie algorytmem DFS określając głębokość odwiedzonych wierzchołków i następnie wyznaczenie wartości funkcji low. Złożoność algorytmu: $O(|V| + |E|)$.

2. **Procedura 1.a:**

   Przeszukiwanie w głąb, z pamięcią dwóch poprzednich wierzchołków. Dla każdego odwiedzanego wierzchołka sprawdzam, czy poprzednie dwa oraz bieżący spełniają założenie. Złożoność czasowa przeszukiwania: $O(|V_p| + |E_p|)$. Aby sprawdzić czy usunięcie wierzhołków pozostawia graf G' spójnym, nalezy wykonać przeszukanie grafu od ostatniego sąsiada pierwszego usuniętego wierzchołka do ostatniego sąsiada drugiego usuniętego wierzchołka. Jeżeli istnieje taka ścieżka graf jest spójny. Złożoność obliczeniowa procedury: $O((|V_p| + |E_p|)^2)$.

3. **Procedura 1.b:**

   Algorytm Dijkstry dla wyznaczenia najkrótszej ścieżki z pojedynczego źródła w grafie. Złożoność algorytmu Dijkstry: $O(|E_p| + |V_p| * log|V_p|)$.

4. **Procedura 1.c:**

   Posortowanie wierzchołków wg. odległości przypisanych w pkt. 2. (np. Quicksort o przeciętnej złożoności $O(|V_p| * log|V_p|)$). Następnie przypisanie wierzchołkom odpowiednich wag ($O(|V_p|)$).

5. **Procedura 2:**

   Zachłanne kolorowanie grafu. Odwiedzam wierzchołki w kolejności ustalonej w pkt. 3., dla każdego wierzchołka sprawdzam wszystkich sąsiadów aby ustalić pierwszy "wolny" kolor. Dla grafu pełnego jest to koszt $O(|V_p|^2)$.

6. **P**rocedura 3: Jeżeli sąsiadujące dwa grafy spójne otrzymane z usunięcia wierzchołków rozspajających w pkt. 1., można połączyć bez zmiany kolorowania któregoś z grafów - takie połączenie jest o stałym koszcie obliczeniowym. Jeżeli wymagana jest zmiana kolorowania któregoś z podgrafów jest to koszt $O(|V_p|)$

Sumując złożoności procedur, algorytm zaproponowany w sekcji 2., jest o wielomianowej złożoności czasowej. Prodecury 1a, 1b, 1c, 2, oraz 3 powtarzane są dla każdego spójnego podgrafu grafu G otrzymanego z usunięcia wierzchołków rozspajających. Zakładając że powstanie k takich podgrafów otrzymujemy złożoność obliczeniową:

$$O(|V| + |E|) + \sum_{i=0}^{k} O(2|V_i|^2 + |V_i|(2|E_i| + log|V_i| + 2) + |E_i|^2) \tag{3}$$

ponieważ: $O(|V_i|(2|E_i| + log|V_i| + 2) + |E_i|^2)) \leqslant O(2|E_i|^2)$, złożoność obliczeniową możemy ograniczyć przez:

$$O(|V| + |E|) + \sum_{i=0}^{k} O(|V_i|^2 + |E_i|^2) \tag{4}$$

gdzie każdy z podrafów i jest o liczbie wierzchołków mniejszej niż n, natomiast $k \leqslant n$. Dla grafu pełnego liczba krawędzi jest równa $|V| * (|V| - 1)/2$ - więc $O(|E|)$ możemy z ograniczyć przez $O(|V|^2)$:

$$O(|V| + |E|) + |V| * O(|V|^4 + |V|^2) \tag{5}$$

Ostatecznie, algorytm jest o złożoności obliczeniowej $O(|V|^5)$.

4 Bibliografia

1. Bryant V., *Aspekty kombinatoryki, s. 144-155*, WNT, Warszawa 1997.
2. *Graph coloring*, en.wikipedia.org.
3. *Greedy algorithm*, en.wikipedia.org.
4. *Dijkstra algorithm*, en.wikipedia.org.
5. *Breadth first search*, en.wikipedia.org.
6. *Quicksort*, en.wikipedia.org.