ROCZNIKI POLSKIEGO TOWARZYSTWA MATEMATYCZNEGO
Seria III: MATEMATYKA STOSOWANA XIX (1982)

Ministerstwo Nauki i Szkolnictwa Wyższego

MAREK KUBALE (Gdańsk)

Analiza efektywności algorytmów kolorowania grafów

(*Praca wpłynęła do Redakcji 15.02.1979*)

Praca poświęcona jest analizie najbardziej znanych algorytmów kolorowania wierzchołków grafów z punktu widzenia ich złożoności obliczeniowej i dobroci kolorowania. Rozpatrzono dwa algorytmy sekwencyjne LF i SL, dwa algorytmy sekwencyjne z wymienianiem kolorów LFI i SLI, algorytm kolorowania par CP oraz uproszczony algorytm maksymalnych zbiorów niezależnych AMIS. Dla każdego algorytmu oszacowano czas potrzebny na jego wykonanie w zależności od rozmiaru grafu oraz podano liczbę kolorów wymaganych w najgorszym przypadku danych. Podano również wnioski płynące z doświadczeń komputerowych związanych z implementacją większości rozważanych metod na maszynie Odra 1204.

1. Wstęp. Metody kolorowania grafów znajdują zastosowanie w wielu różnorodnych problemach techniki i zarządzania. Dla przykładu, następujące zagadnienia mogą być sprowadzone do problemu kolorowania wierzchołków odpowiedniego grafu: podział układu elektronicznego na warstwy, znajdowanie minimalnej liczby testów potrzebnych do lokalizacji uszkodzenia układu logicznego, projektowanie kodów wykrywających błędy w urządzeniach transmisji danych, minimalizacja liczby wykonawców prac o ustalonym harmonogramie, sporządzanie rozkładów zajęć szkolnych. We wszystkich tych zagadnieniach chodzi o pokolorowanie grafu szybko i oszczędnie, tzn. przy użyciu możliwie najmniejszej liczby barw. Wymagania te są na ogół sprzeczne.

Obecnie wszystkie problemy kombinatoryczne dzieli się na „łatwe" i „trudne". Do pierwszej klasy, zwanej krótko P, należą te, dla których istnieją algorytmy o wielomianowym czasie obliczeń (ściślej — o czasie obliczeń ograniczonym przez wielomian zmiennej rozmiaru problemu), do drugiej zaś te, dla których nie udało się do dzisiaj znaleźć żadnej takiej metody obliczeniowej. Wśród zagadnień tej drugiej klasy wyróżnić można obszerną rodzinę problemów tzw. NP-zupełnych [9] (zob. też [18]), które są między sobą równoważne w następującym sensie: jeżeli dla jednego z problemów tej rodziny uda się znaleźć algorytm wielomianowy, to dla każdego problemu NP-zupełnego będzie można skonstruować tego typu algorytm. Przypuszcza się, że dla żadnego problemu NP-zupełnego nie istnieje wielomianowy algorytm rozwiązywania. Niestety zagadnienie optymalnego kolorowania grafu należy do tej rodziny.

Głębsza analiza problemów NP-zupełnych wykazała, że nie są one wszystkie jednakowej trudności. Na podstawie jakości rozwiązań przybliżonych generowanych

przez najlepsze algorytmy wielomianowe (tj. takie, które dla poszczególnych zagadnień gwarantują rozwiązania najbardziej zbliżone do optymalnych w najgorszym przypadku danych) rodzinę problemów NP-zupełnych dzielimy na trzy następujące grupy:

1. Problemy, które mają wielomianowe algorytmy gwarantujące rozwiązania przybliżone z dowolnie zadaną dokładnością względną $\varepsilon > 0$. Przypomnijmy, że dla ustalonych danych rozwiązanie o wartości r nazywamy *ε-przybliżonym*, jeśli $|r-r_0|/|r_0| \leqslant \varepsilon$, gdzie r_0 jest wartością rozwiązania optymalnego. Do grupy tej należą między innymi problem plecakowy i niektóre przypadki zagadnienia kolejnościowego.

2. Problemy, dla których najlepsze algorytmy wielomianowe gwarantują roz wiązania z ustaloną, ale nie zależną od nas, dokładnością. Do grupy tej zaliczamy np. problem pakowania (gwarantowana tolerancja 22%), metryczny problem komiwojażera (tolerancja 50%), problem minimalnego drzewa Steinera (z metryką prostokątną — 50%), problem operatora dźwigu (tolerancja 80%).

3. Problemy, dla których istniejące algorytmy wielomianowe nie gwarantują żadnej skończonej dokładności, tzn. nie istnieje żadna liczba $\varepsilon > 0$, dla której jakiś algorytm gwarantowałby rozwiązanie ε-przybliżone. Do grupy tej zaliczamy wszystkie pozostałe problemy, w szczególności problem znajdowania maksymalnej kliki grafu, pewne problemy pokrycia, programowanie całkowitoliczbowe i niestety ... problem kolorowania grafu. Dla niektórych z tych problemów udowodniono, że nie istnieje liczba $k > 1$, dla której znalezienie rozwiązania nie gorszego niż k razy wartość optymalna byłoby łatwiejsze niż wyznaczenie rozwiązania optymalnego.

Powyższa klasyfikacja problemów ma charakter w pewnym sensie dynamiczny. Nie jest bowiem wykluczone, iż w wyniku stałego postępu w konstrukcji algorytmów przybliżonych niektóre z problemów należących obecnie do grupy 2 lub 3 awansują do grup wcześniejszych. Dotyczy to tylko nielicznych problemów, zwłaszcza zaś tych, które wymagają na wejściu danych liczbowych w postaci wektorów, macierzy itp. (patrz [7]). Zauważmy, że jest bardzo mało prawdopodobne aby problem kolorowania grafu przeszedł do pierwszej grupy, gdyż w świetle [6] pokolorowanie grafu liczbą kolorów nie przekraczającą w dwójnasób minimalnej możliwej jest tak samo trudne jak wyznaczenie rozwiązania optymalnego. Oznacza to, że problem kolorowania grafu przejdzie do grupy 1 jedynie wtedy, gdy wszystkie problemy NP-zupełne znajdą się w tej grupie (tzn., gdy P = NP).

Problemy związane z analizą algorytmów pojawiają się wówczas, gdy dla danego zagadnienia dysponujemy już całą rodziną metod jego rozwiązania. Powstają wtedy naturalne pytania: która metoda jest najlepsza oraz czy są szanse znalezienia jeszcze lepszych algorytmów. Niniejsza praca jest przeglądem najbardziej znanych algorytmów w rodzinie metod kolorowania wierzchołków grafów oraz próbą odpowiedzi na postawione wyżej pytania. Po wprowadzeniu podstawowej terminologii oraz oszacowaniu złożoności obliczeniowej problemu optymalnego kolorowania, w kolejnych punktach badamy złożoność przybliżonych algorytmów wielomianowych

wyznaczoną poprzez analizę najgorszego przypadku danych. Dla niektórych algorytmów podajemy przykłady grafów, dla których rozważany algorytm działa szczególnie wolno lub używa szczególnie dużo kolorów. Analizę teoretyczną zamykamy uwagami na temat rezultatów badań empirycznych powyższych metod. Z punktu widzenia praktyki jest to istotny etap w badaniach nad efektywnością algorytmów i ich konkretnych realizacji.

2. Podstawowe definicje i oznaczenia. Na wstępie podamy podstawowe oznaczenia i definicje teorii grafów, których używać będziemy w dalszych punktach pracy. W artykule rozważać będziemy grafy *zwykłe*, tj. symetryczne grafy skończone bez pętli i bez krawędzi wielokrotnych. Zbiór *wierzchołków* grafu G oznaczamy przez V, a zbiór *krawędzi* przez E. Krawędź utożsamiamy z dwójką *incydentnych* z nią wierzchołków, powiedzmy v_i, $v_j \in V$, i piszemy $\{v_i, v_j\}$. Liczbę wierzchołków nazywamy *rzędem* grafu i oznaczamy przez n, $n = |V|$, a liczbę krawędzi oznaczamy przez m, $m = |E|$. Liczbę $d = 2m/n(n-1)$ nazywamy *gęstością* grafu $G = (V, E)$.

Mówimy, że dwa wierzchołki $v_i, v_j \in V$ są *sąsiednie* w grafie $G = (V, E)$, jeżeli $\{v_i, v_j\} \in E$. Wierzchołki, które nie są sąsiednie nazywamy *niesąsiednimi*. *Stopniem* $\deg(v)$ wierzchołka v jest liczba wierzchołków sąsiednich z v lub inaczej liczba incydentnych z nim krawędzi.

Podgraf grafu $G = (V, E)$ *indukowany* przez zbiór $V' \subset V$ oznaczamy przez $G|V' = (V', E')$, gdzie E' jest zbiorem tych krawędzi należących do E, których oba końce należą do zbioru V'.

Niech będzie dany graf G i liczba naturalna k. Mówimy, że G jest *k-barwny*, jeżeli jego wierzchołki można pomalować k różnymi barwami w taki sposób, że żadne wierzchołki sąsiednie nie są pokolorowane tą samą barwą. Mówimy wtedy również, że istnieje *k-pokolorowanie* grafu G. Najmniejszą liczbę k, dla której graf G jest k-barwny nazywamy *liczbą chromatyczną* grafu G i oznaczamy przez $\chi(G)$, zaś graf G nazywamy *k-chromatycznym*. Zatem k-pokolorowanie grafu $G = (V, E)$ jest równoważne wyznaczeniu rodziny podzbiorów

(1) $$\{V_1, \ldots, V_k\}$$

takiej, że

(2a) $$V_i \subseteq V, \quad i = 1, \ldots, k,$$

(2b) $$V_i \cap V_j = \emptyset, \quad i \neq j,\ i, j = 1, \ldots, k,$$

(2c) $$\text{jeśli } v, v' \in V_i, \text{ to } \{v, v'\} \notin E, \quad i = 1, \ldots, k.$$

Jeżeli ponadto

(3) $$\bigcup_{i=1}^{k} V_i = V,$$

to pokolorowanie (1) nazywamy *pełnym*. W przeciwnym razie pokolorowanie grafu nazywamy *częściowym*. Pokolorowanie pełne takie, że $k = \chi(G)$ nazywamy *optymalnym*. Istnieje szereg algorytmów kolorowania grafów zarówno dokładnych (optymalnych) [3], [5], [16], [20], [21] jak i przybliżonych (suboptymalnych) [4], [14], [22], [23]. Przez $A(G)$ oznaczamy liczbę kolorów użytych do pomalowania grafu G

metodą A. Z każdą taką metodą związać można *funkcję dobroci* $f_A: N \to R_+$ zdefiniowaną następująco: $f_A(n) = \max\{A(G)/\chi(G): G$ ma n wierzchołków$\}$. Funkcja ta charakteryzuje jakość rozwiązań generowanych przez algorytm A dla najgorszego przypadku danych. Zauważmy, że dla każdego algorytmu kolorowania $A, f_A(n) \leqslant n$.

Podzbiór V_i wierzchołków grafu G spełniających warunek (2c) nazywamy *niezależnym*. Zbiór niezależny nazywamy *maksymalnym*, jeżeli nie istnieje jego nadzbiór właściwy, który jest niezależny.

Problem optymalnego pomalowania grafu można również sformułować w następujący sposób. Rozważmy graf $G = (V, E)$. Definiujemy funkcję kolorowania $h: V \to N$ taką, że $h(v_i) \neq h(v_j)$, gdy $\{v_i, v_j\} \in E$. W zbiorze H wszystkich funkcji kolorowania szukamy funkcji h_0, dla której

$$\min_{h \in H} \max_{v \in V} h(v) = \max_{v \in V} h_0(v) = \chi(G). \tag{4}$$

3. Złożoność obliczeniowa problemu optymalnego kolorowania grafu. Naczelnym pytaniem w dziedzinie kolorowania grafów jest pytanie o złożoność obliczeniową problemu optymalnego pokolorowania dowolnego grafu. Niestety nie jest znany rząd złożoności obliczeniowej, a jedynie dolne i górne oszacowanie liczby działań potrzebnych dla rozwiązania tego problemu. Złożoność obliczeniowa kolorowania, tak jak i innych problemów kombinatorycznych, jest ściśle związana ze sposobem reprezentacji grafu (np. w pamięci komputera). O wpływie sposobu reprezentacji grafu na efektywność metod obliczeniowych pisze Sysło w pracy [18]. W niniejszym artykule przyjmiemy listowy sposób pamiętania struktury grafu, ponieważ umożliwia on uzyskanie bardziej efektywnych algorytmów w porównaniu ze sposobem macierzowym. Założenie to implikuje od razu dolne oszacowanie liczby koniecznych działań postaci

$$\Omega(m+n) \leqslant \varphi(m, n), \tag{5}$$

gdzie $\varphi(m, n)$ oznacza złożoność obliczeniową problemu optymalnego pokolorowania grafu G o m krawędziach i n wierzchołkach([1]). Oszacowanie (5) wynika stąd, że z jednej strony każda krawędź jest tu istotna i musi być sprawdzona co najmniej raz, z drugiej zaś każdy wierzchołek musi mieć nadany jakiś kolor. Zauważmy, że dla grafów dwudzielnych $\varphi(m, n) = O(m+n)$. Równość ta wynika stąd, że każdy graf dwudzielny może być optymalnie pomalowany w czasie $O(m+n)$ przy okazji trawersowania tego grafu metodą pogłębiania. Zatem grafy G takie, że $\chi(G) \leqslant 2$ mogą być kolorowane za pomocą algorytmu optymalnego z dokładnością do stałej proporcjonalności.

Odnotujmy, że gdyby przyjąć, iż graf zapisany jest za pomocą macierzy sąsiedztwa, to minimalna liczba operacji wzrośnie do $\Omega(n^2)$. Zgodnie bowiem z tezą Aanderaa–Rosenberga udowodnioną przez Rivesta i Vuillemina [15] zbadanie każdej nietrywialnej i monotonicznej własności dowolnego grafu (a taką jest niemożność pokolorowania grafu G k kolorami) wymaga co najmniej $n^2/16$ operacji podstawowych przy macierzowym sposobie zapisu grafu.

([1]) Objaśnienia symboli $\Omega(\cdot)$ i $O(\cdot)$ znaleźć można w [18].

Obecnie, za Lawlerem[12], podamy górne oszacowanie złożoności obliczeniowej problemu kolorowania, które oparte jest na następującym rozumowaniu. Zauważmy, że jeżeli graf G jest k-chromatyczny, to istnieje podział jego wierzchołków na k zbiorów niezależnych taki, że przynajmniej jeden ze zbiorów jest maksymalny. Wynika to stąd, że jeżeli uda się nam znaleźć jakieś pokolorowanie optymalne, to dowolny kolor będzie albo maksymalnym zbiorem niezależnym, albo da się powiększyć do takiego. A zatem, jeżeli zbiór $V' \subseteq V$ jest niepusty, to istnieje maksymalny zbiór niezależny $S \subset V'$ taki, że $\chi(G|V') = 1+\chi(G|(V'-S))$. Istnieje skończona liczba maksymalnych zbiorów niezależnych podgrafu $G|V'$. Minimalizując po nich wszystkich otrzymujemy

$$\chi(G|V') = 1+\min_{S\subset V'}\chi(G|(V'-S)), \quad \text{gdzie } V' \neq \emptyset. \tag{6}$$

Zatem liczba operacji potrzebnych do wyznaczenia liczby chromatycznej jest równa liczbie działań potrzebnych do rozwiązania równości (6). Oszacujmy tę liczbę dla najgorszego przypadku danych. Załóżmy, że dla ustalonego V' znaleźliśmy już $\chi(G|V'')$ po wszystkich $V'' \subset V'$. Wtedy czas wymagany dla obliczenia $\chi(G|V')$ jest proporcjonalny do liczby maksymalnych zbiorów niezależnych podgrafu $\chi(G|V')$ plus czas ich wygenerowania. Znany, najbardziej efektywny algorytm generowania wszystkich maksymalnych zbiorów niezależnych grafu G wymaga $O(nml)$ operacji podstawowych, gdzie m jest liczbą krawędzi, n — liczbą wierzchołków, a l — liczbą maksymalnych zbiorów niezależnych danego grafu [19]. W najgorszym przypadku liczba l jest równa $3^{n/3}$. Z drugiej strony wystarczy odwiedzić 2^n wierzchołków tzw. drzewa podgrafów grafu G (szczegóły patrz [20]). Sumując po wszystkich możliwych $V' \subseteq V$ znajdujemy, że ogólny czas wymagany dla rozwiązania (6) jest ograniczony funkcją

$$\sum_{i=0}^{n}\binom{n}{i}m_i i 3^{i/3} < mn\sum_{i=0}^{n}\binom{n}{i}3^{i/3} = mn(1+\sqrt[3]{3})^n,$$

gdzie m_i jest maksymalną liczbą krawędzi w zbiorze odpowiednich podgrafów grafu G opartych na i wierzchołkach.

Reasumując otrzymujemy, że złożoność obliczeniowa problemu optymalnego pokolorowania grafu o m krawędziach i n wierzchołkach spełnia nierówności:

$$\Omega(m+n) \leqslant \varphi(m,n) \leqslant O(mn2.44^n).$$

4. Algorytmy sekwencyjne. *Algorytm sekwencyjny* zdefiniowany jest następująco. Niech wierzchołki grafu będą ustawione w ciąg $v_1, \ldots, v_n$. Wówczas pomaluj v_1 kolorem 1, a każdy następny wierzchołek v_i kolorem o możliwie najmniejszym numerze, tzn. z zachowaniem warunku, że wierzchołki sąsiednie otrzymują różne barwy. Oczywiście kolorowania otrzymywane tą drogą w istotny sposób zależą od przyjętego uszeregowania wierzchołków. Na przykład każdy graf może być optymalnie pokolorowany za pomocą algorytmu sekwencyjnego. Wystarczy jedynie znać właściwą kolejność kolorowania. Lecz znalezienie właściwej kolejności w zbiorze $n!$ możliwych wcale nie jest łatwe. Istnieją dwa podstawowe sposoby uporządkowania wierzchołków będące źródłem dwóch różnych algorytmów sekwencyjnych.

W metodzie *pierwszy-największy*, symbolicznie LF (ang. largest first), wierzchołki są porządkowane w kolejności nierosnących stopni, a następnie odbywa się właściwe kolorowanie sekwencyjne. Algorytm ten został po raz pierwszy podany przez Welsha i Powella [22] w nieco odmiennej lecz równoważnej postaci. W najgorszym przypadku wymaga on liczby kolorów określonej wzorem

$$\chi(G) \leqslant \alpha(G) = \max_{1 \leqslant i \leqslant n} \min \{i, \deg(v_i)+1\}, \tag{8}$$

lecz na ogół liczba kolorów użytych jest znacznie mniejsza od $\alpha(G)$.

Obecnie bardziej szczegółowo opiszemy algorytm LF używając konstrukcji pseudoalgolowskich. W tym celu przyjmiemy, że struktura grafu zapamiętana jest w tablicy całkowitej $G\,[1:2m]$, której kolejne grupy elementów odpowiadają wszystkim wierzchołkom sąsiednim odpowiednio z $v_1, \ldots, v_n$. Ściślej: dla $i = 1, \ldots, n$ oraz $j = 1, \ldots, 2m$ G_j jest numerem wierzchołka sąsiedniego z v_i, jeżeli $s_{i-1}+1 \leqslant j \leqslant s_i$, gdzie $s_0 = 0$ a $s_i = s_{i-1}+\deg(v_i)$ $(i = 1, \ldots, n)$. Algorytm LF składa się z trzech etapów. W pierwszym etapie obliczamy stopnie wszystkich wierzchołków. Łatwo zauważyć, że jeżeli $\mathrm{Deg}\,[1:n]$ jest tablicą zerową, to stopnie wierzchołków otrzymamy kosztem $2m$ dodawań w wyniku wykonania następującej instrukcji:

for $j := 1, 2, \ldots, 2m$ **do**
  $\mathrm{Deg}\,[G_j] := \mathrm{Deg}\,[G_j]+1$;

W drugim etapie algorytmu porządkujemy wierzchołki w kolejności nierosnących stopni. W tym celu zauważmy, że znane są a *priori* klucze, według których przeprowadzamy sortowanie, gdyż dla każdego i mamy $\deg(v_i) \in \{0, \ldots, n-1\}$. Zatem jeżeli będziemy znali liczbę wierzchołków stopnia $0, 1, \ldots, n-1$, to stąd łatwo wyznaczymy końcową pozycję jaką może zająć dany wierzchołek w nowym uporządkowaniu. Wymagane uporządkowanie przedstawimy w postaci tablicy Newseq $[1:n]$, gdzie Newseq_i jest numerem i-tego wierzchołka w nowym uporządkowaniu. Jeżeli Kosz $[0:n]$ jest początkowo tablicą zerową, to naszkicowaną ideą realizuje następujący fragment programu:

for $i := 1, 2, \ldots, n$ **do**
  $\mathrm{Kosz}\,[\mathrm{Deg}_i] := \mathrm{Kosz}\,[\mathrm{Deg}_i]+1$;
for $i := n-1, i-1$ **while** $\mathrm{Kosz}_{i+1} < n$ **do**
  $\overline{\mathrm{Kosz}}_i := \mathrm{Kosz}_i := \mathrm{Kosz}_i+\mathrm{Kosz}_{i+1}$;
comment Kosz_i jest końcową pozycją grupy wierzchołków stopnia i. Cel wprowadzenia tablicy $\overline{\mathrm{Kosz}}$ wyjaśniony zostanie w dalszej części pracy;
for $i := 1, 2, \ldots, n$ **do**
begin
  $\mathrm{Newseq}\,[\mathrm{Kosz}\,[\mathrm{Deg}_i]] := i$;
A: ; **comment** instrukcja pusta, której cel zostanie wyjaśniony dalej;
  $\mathrm{Kosz}[\mathrm{Deg}_i] := \mathrm{Kosz}[\mathrm{Deg}_i]-1$
end i

Widzimy, że dla uzyskania żądanego uporządkowynia należy wykonać n odejmowań

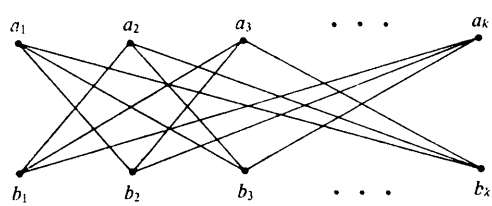
i najwyżej $2n$ dodawań. Zatem drugi etap algorytmu może być zrealizowany w czasie proporcjonalnym do liczby wierzchołków grafu.

Aby zbadać złożoność ostatniego etapu algorytmu przyjmiemy, że z każdym wierzchołkiem v_i związany jest n-elementowy wektor charakterystyczny obrazujący stan pokolorowania wierzchołków sąsiednich z v_i. Mianowicie, k-ty element tego wektora jest równy 0, gdy żaden z sąsiadów v_i nie został pomalowany kolorem k, a 1 w przypadku przeciwnym. Przez *zastrzeżenie* koloru k rozumieć będziemy wstawienie jedynki na k-tą pozycję wektora charakterystycznego. Przyjmiemy również, że żaden kolor nie został zastrzeżony przed rozpatrywaniem kolejnego wierzchołka. Przy tych założeniach procedurę kolorowania można opisać za pomocą następującej instrukcji:

for $i := 1, 2, \ldots, n$ **do**
  begin
  A: **for** każdego sąsiada w wierzchołka nr Newseq_i **do** zastrzeż kolor wierzchołka w;
  B: **for** $k := 1, 2, \ldots, \min\{i, \deg(v_{\mathrm{Newseq}[i]})+1\}$ **do**
    if kolor k nie jest zastrzeżony **then**
      begin
      $V_k \leftarrow V_k \cup \{v_{\mathrm{Newseq}[i]}\}$;
      comment V_k jest zbiorem wierzchołków pomalowanych k-tym kolorem;
    go to e
  end k;
  e: **end** i

Przeanalizujmy złożoność tego etapu algorytmu. W i-tym kroku zastrzeżenie kolorów wymaga $\deg(v_{\mathrm{Newseq}_i})$ podstawień, a pomalowanie wierzchołka wymaga jednego podstawienia poprzedzonego nie więcej niż $\deg(v_{\mathrm{Newseq}_i})+1$ porównaniami. Ponieważ pętle (²) A i B wykonywane są dla każdego wierzchołka jeden raz, więc łącznie mamy $\sum_{i=1}^{n}\left(\deg(v_i)+1\right) = 2m+n$ podstawień i najwyżej tyle samo porównań. Zatem ostatecznie algorytm LF wymaga $O(m+n)$ operacji podstawowych.

Metoda LF daje bardzo złe pokolorowania już w przypadku grafów G_k, $k \geqslant 3$ o liczbie chromatycznej równej 2 [8]. Przykład takiego grafu pokazano na rysunku 1.



Rys. 1. Graf G_k

(²) Tak dalej nazywać będziemy instrukcje „dla".

Jeżeli wierzchołki uporządkowane są w kolejności $a_1, b_1, a_2, b_2, \ldots, a_k, b_k$, to algorytm LF pomaluje każdą parę nowym kolorem. Zatem $\mathrm{LF}(G_k) = k$ chociaż $\chi(G_k) = 2$ przy $|V_k| = 2k = O(k)$, czyli funkcja dobroci spełnia $f_{\mathrm{LF}}(n) = \Omega(n)$.

W metodzie *ostatni-najmniejszy*, symbolicznie SL (smallest last) wierzchołki są ustawiane w nieco bardziej skomplikowany sposób. Mianowicie, jako v_n brany jest wierzchołek o najmniejszym stopniu w grafie G, a następnie dla każdego $i = n-1, \ldots, 2, 1$ jako v_i wybieramy wierzchołek, który ma minimalny stopień w podgrafie $G|(V-\{v_n, \ldots, v_{i+1}\})$. Proces ten można interpretować jako proces redukcji grafu polegającej na sukcesywnym usuwaniu wierzchołków (wraz z incydentnymi krawędziami) o aktualnie najmniejszym stopniu [4]. Po takim uporządkowaniu wierzchołki są kolorowane tak jak w metodzie LF. Łatwo udowodnić, że

$$\chi(G) \leqslant \beta(G) = \max_{1 \leqslant i \leqslant n} \overline{\deg}(v_i) + 1, \tag{9}$$

gdzie $\overline{\deg}(v_i)$ jest stopniem $\deg(v_i)$ w chwili usuwania wierzchołka v_i z podgrafu $G|(V-\{v_n, \ldots, v_{i+1}\})$.

Nasuwa się pytanie, które oszacowanie liczby chromatycznej (8) czy (9) jest bardziej dokładne. Matula udowodnił, że oszacowanie (9) związane z kolejnością ostatni-najmniejszy jest najlepszym możliwym pośród wszystkich $n!$ ciągów wierzchołków dowolnego grafu G (por. [14]). A zatem $\alpha(G) \geqslant \beta(G)$ dla każdego grafu G. Mimo to oszacowanie $\beta(G)$ jest na ogół bardzo złe nie tylko w odniesieniu do liczby chromatycznej grafu G, lecz nawet w stosunku do liczby kolorów faktycznie użytych do jego pokolorowania metodą SL. Badania eksperymentalne ([11], [14]) wskazują, że dla grafów dużych i gęstych $\beta(G) \geqslant 2\chi(G)$.

Obecnie pokażemy w jaki sposób algorytm SL może być wykonany w czasie $O(m+n)$. Aby efektywnie ustawić wierzchołki w kolejności SL zauważmy, że ostatni wierzchołek w dowolnej kolejności LF może być ostatnim wierzchołkiem w pewnej kolejności SL. Po usunięciu tego wierzchołka i sprowadzeniu pozostałych wierzchołków podgrafu do kolejności LF ponownie ostatni wierzchołek podzbioru może być przedostatnim wierzchołkiem w sekwencji SL. Kontynuując ten proces otrzymamy żądane uporządkowanie. Jeżeli w miejsce instrukcji pustej po etykiecie A we fragmencie obliczającym kolejność LF wstawimy instrukcję

A: $\mathrm{Poz}_i := \mathrm{Kosz}[\mathrm{Deg}_i]$;

to odpowiedni fragment algorytmu, stanowiący niejako kontynuację porządkowania LF, będzie można zapisać następująco:

```
for i := n, n-1, ..., 3 do
  begin
    Kosz[Deg[Newseq_i]] := i-1;
  B: for każdego v_j sąsiedniego z wierzchołkiem nr Newseq_i do
       if Poz_j < i then
         begin
           comment najpierw przestawiamy odpowiednie pary wierzchołków;
```

```
        Newseq[Poz_j]: = Newseq[Kosz[Deg_j]];
        Newseq[Kosz[Deg_j]]: = j;
        Poz[Newseq[Poz_j]]: = Poz_j; Poz_j: = Kosz[Deg_j];
        comment obecnie aktualizujemy końcową pozycję grupy wierzchołków
        stopnia deg(v_j) oraz redukujemy stopień wierzchołka v_j;
        Kosz[Deg_j]: = Kosz[Deg_j]-1;
        Deg_j: = Deg_j-1
    end B
end i
```

Pewnego komentarza wymaga fakt zakończenia obliczeń przy wartości zmiennej sterującej $i = 3$. Otóż, jak łatwo sprawdzić, dla każdego grafu rzędu 2 i 3 spełniona jest własność: każde uporządkowanie wierzchołków LF jest zarazem kolejnością SL. Jednakże poczynając od $n = 4$ istnieją grafy, które nie mają powyższej własności.

Obecnie przeanalizujemy złożoność obliczeniową metody SL. W powyższym fragmencie porządkowania wykonanie pętli B wymaga $\deg(v_{\mathrm{Newseq}_i})$ porównań i $2\deg(v_{\mathrm{Newseq}_i})$ odejmowań. Pętla ta jest wykonywana $n-2$ razy dla $n-2$ różnych wierzchołków. Ponieważ każde wykonanie pętli jest poprzedzone odejmowaniem, więc łącznie mamy co najwyżej $2m$ porównań i $2m+n$ odejmowań. Skoro wstępne porządkowanie wierzchołków (otrzymanie kolejności LF) wymaga wykonania $O(n)$ operacji podstawowych, to cały etap tworzenia kolejności SL może być zrealizowany w czasie $O(m+n)$. Do powyższego należy dodać $O(m)$ operacji potrzebnych do obliczenia stopni wierzchołków i $O(m+n)$ operacji dla właściwego pomalowania grafu. Zatem ostatecznie algorytm SL wymaga $O(m+n)$ operacji podstawowych.

Burlaga [4] zauważył, że grafy Mycielskiego dają się optymalnie pomalować metodą SL. Można pokazać, że metoda ta zastosowana do kolorowania grafów płaskich daje $O(n)$ algorytm, który używa co najwyżej 6 kolorów. Odnotujmy tutaj, że najlepszym znanym sposobem kolorowania takich grafów jest $O(n\log n)$ metoda „partiowa" (ang. batching method) przedstawiona w [13], która używa najwyżej 5 barw wobec 4 możliwych (co ostatecznie udowodniono w [1], [2]). Mimo to w ogólności istnieje ciąg grafów G_k o $O(k)$ wierzchołkach taki, że $\mathrm{SL}(G_k) = k$, chociaż $\chi(G_k) = 3$ [8].

5. Algorytmy sekwencyjne z wymienianiem kolorów. Istnieje prosta metoda poprawiania rozwiązań uzyskiwanych przez algorytmy sekwencyjne podana przez Matulę [14], a następnie uogólniona przez Johnsona [8]. Metoda ta oparta jest na następującym spostrzeżeniu. Jeżeli w i-tym kroku kolorowania wierzchołek v_i winien otrzymać nowy kolor, to być może można zmienić barwy wierzchołkom już pomalowanym tak, aby z jednej strony zachować warunek, iż żadna krawędź nie łączy wierzchołków jednobarwnych, z drugiej zaś, by któryś z wcześniejszych kolorów uczynić dostępnym dla v_i. Jeśli tak, to dokonujemy wymiany kolorów, a następnie malujemy v_i i dalsze wierzchołki danego ciągu. Zatem algorytm sekwencyjny z wymienianiem kolorów działa tak samo jak zwykły algorytm sekwencyjny z wyjątkiem przypadku, gdy normalnie należałoby wprowadzić nowy kolor.

Obecnie opiszemy dokładnie przypadek szczególny metody Johnsona, który nazwiemy *naiwnym wymienianiem* lub krótko *wymienianiem* kolorów. Metoda naiwnego wymieniania zdefiniowana jest następująco (por. [11]). Jeżeli wierzchołek v_i nie może być pomalowany żadnym z k, $k > 1$, istniejących już kolorów, to tworzymy zbiór $S \subseteq \{1, \ldots, k\}$ numerów kolorów, z których każdy posiada dokładnie jeden wierzchołek sąsiedni z v_i. Następnie, jeśli dla pewnego $a \in S$ istnieje wierzchołek v_j pomalowany a-tą barwą i sąsiedni z v_i, lecz nie sąsiedni z żadnym wierzchołkiem innego koloru $b \leqslant k$, to v_j otrzymuje kolor b. Umożliwia to pomalowanie v_i kolorem a. Jeżeli natomiast takiego wierzchołka nie ma, to v_i staje się pierwszym wierzchołkiem koloru $k+1$. Z opisu metody wymieniania kolorów wynika, że wymaga ona każdorazowo uprzedniego wyznaczenia zbioru wierzchołków pomalowanych, będących w swoich kolorach jedynymi wierzchołkami sąsiednimi z v_i. Oznaczymy ten zbiór przez S_i. Zbiór S_i może być wyznaczony w czasie $O(\deg(v_i))$. Jeśli znamy już S_i, to wymienianie może być zapisane następująco:

```
A: for każdego v_j ∈ S_i do
    begin
    B:for każdego sąsiada w wierzchołka v_j do
        zastrzeż kolor wierzchołka w;
      zastrzeż kolor a wierzchołka v_j;
      comment k jest liczbą użytych dotąd kolorów;
  C: for b: = 1, 2, ..., k do
        if kolor b nie jest zastrzeżony then
        begin
         V_b ← V_b ∪ {v_j};
         V_a ← V_a ∪ {v_i} − {v_j};
         go to e
        end b
    end; e:
```

Zbadajmy złożoność obliczeniową algorytmów sekwencyjnych z naiwnym wymienianiem kolorów. Przez LFI (LF with interchange) oznaczymy algorytm sekwencyjny z wymienianiem, jeśli wierzchołki zostały uporządkowane w kolejności pierwszy-największy, a przez SLI (SL with interchange) algorytm sekwencyjny z wymienianiem oparty na kolejności ostatni-najmniejszy. Zauważmy, że gdyby w procesie kolorowania nie doszło do żadnej próby wymiany koloru (tzn. graf dałby się pomalować dwoma kolorami), to czas wykonania byłby taki sam jak dla oryginalnych algorytmów sekwencyjnych, a więc $O(m+n)$. Jednakże w najgorszym przypadku próba wymiany kolorów może nastąpić $n-2$ razy, tzn. przy kolorowaniu każdego wierzchołka $v_3, \ldots, v_n$ (np. w przypadku grafów pełnych). Zatem o złożoności obliczeniowej algorytmu sekwencyjnego z wymienianiem decyduje złożoność samego wymieniania. Zbadajmy złożoność tego etapu. Weźmy pod uwagę instrukcje podstawienia. Dla kolejnego wierzchołka v_j pętla B wymaga wykonania $\deg(v_j)$ podstawień plus jedno podstawienie dla zastrzeżenia koloru a. W najgorszym przypadku

operacje te są powtarzane dla każdego $v_j \in S_i$, a więc łącznie należy wykonać co najwyżej następującą liczbę podstawień:

$$\sum_{v_j \in S_i} (\deg(v_j)+1) = \deg(v_i) + \sum_{v_j \in S_i} \deg(v_j) \leqslant \deg(v_i) + \sum_{v_j \in V-\{v_i\}} \deg(v_j) = 2m.$$

W podobny sposób można pokazać, że liczba porównań jest nie większa niż $2m$. Skoro w najgorszym przypadku pętla A jest wykonywana $O(n)$ razy, to ogólna liczba operacji jest ograniczona przez $O(mn)$. Wobec tego złożoność obliczeniowa algorytmów LFI i SLI jest rzędu mn.

Najgorszy przypadek, w sensie jakości rozwiązań, dla metody LFI jest taki sam jak dla algorytmu LF. Jeżeli wierzchołki grafu G_k na rysunku 1 są malowane w kolejności $a_1, b_1, a_2, b_2, \ldots$, to algorytm LFI pomaluje każdą parę nowym kolorem, gdyż żadna wymiana nie będzie mogła mieć tu miejsca. Zatem funkcja dobroci ma postać $f_{\mathrm{LFI}}(n) = \Omega(n)$. Podobnie metoda SLI wykazuje bardzo złe zachowanie w najgorszym przypadku danych, które charakteryzuje się liniową funkcją f_{SLI}. Doświadczenia wykazały, że przeciętnie metody z naiwnym wymienianiem dają znaczne zmniejszenie liczby kolorów w porównaniu z metodami LF i SL sięgające nawet 10 barw dla grafów losowych rzędu $n = 200$ i gęstości $d \geqslant 0.5$ [11].

6. Algorytm kolorowania par. *Algorytm kolorowania par*, symbolicznie CP (colouring pairs), jest pewną modyfikacją algorytmu sekwencyjnego zaproponowaną przez Wooda [23]. Polega on na uporządkowaniu wszystkich par wierzchołków, a następnie sekwencyjnym ich kolorowaniu. Algorytm wymaga wprowadzenia tzw. *macierzy podobieństw* wierzchołków grafu G, $\mathbf{S} = [s_{ij}]$, $i, j = 1, \ldots, n$. Jeżeli graf zapisany jest za pomocą macierzy sąsiedztwa wierzchołków $\mathbf{A} = [a_{ij}]$, $i, j = 1, \ldots, n$, to

$$s_{ij} = \begin{cases} -1, & \text{gdy } i = j \text{ lub } a_{ij} = 1, \\ \sum_{k=1}^{n} a_{ik} a_{kj} & \text{w przeciwnym razie.} \end{cases}$$

Zatem stopień podobieństwa dwóch wierzchołków niesąsiednich v_i i v_j jest równy liczbie wierzchołków v_k jednocześnie sąsiednich z v_i i v_j lub inaczej, jest równy liczbie różnych dróg długości 2 łączących v_i z v_j. Po obliczeniu macierzy $\mathbf{S}$ wszystkie $n(n-1)/2$ pary wierzchołków są porządkowane według nierosnących podobieństw. Dalej algorytm analizuje po kolei wszystkie pary o nieujemnych podobieństwach w następujący sposób:

1. Jeżeli v_i i v_j są już pomalowane, to przejdź do następnej pary.

2. Jeżeli jeden z nich, powiedzmy v_i, ma już r-ty kolor, a v_j jest niepomalowany, to:

(a) Jeśli stopień $\deg(v_j)$ jest mniejszy od liczby kolorów dotąd użytych, to przejdź do następnej pary.

(b) Spróbuj dołączyć v_j do koloru r; jeśli jest to niemożliwe, przejdź do następnej pary.

3. Jeżeli ani v_i, ani v_j nie są pomalowane, to:

(a) Jeśli stopnie obu wierzchołków są mniejsze od liczby kolorów użytych, to przejdź do następnej pary.

(b) Pomaluj v_i i v_j kolorem, być może nowym, o możliwie najniższym numerze.

Algorytm CP jest metodą częściową. Oznacza to, że zwykle po jego zakończeniu niektóre wierzchołki pozostają niepokolorowane. Wierzchołki niepokolorowane, które były pomijane w podpunktach (a), nazwiemy wierzchołkami *bezbarwnymi pierwszego rodzaju.* Do grupy tej zaliczamy również wierzchołki początkowo pominięte w podpunkcie 2(b), które następnie były analizowane w podpunktach (a). Pozostałe wierzchołki niepokolorowane nazwiemy *bezbarwnymi drugiego rodzaju.* Niech k oznacza liczbę kolorów użytych dla częściowego pokolorowania grafu G. Zauważmy, że skoro stopnie wierzchołków bezbarwnych pierwszego rodzaju są mniejsze od k, to wierzchołki te mogą być pomalowane po zakończeniu algorytmu CP bez wprowadzania nowych kolorów. Zauważmy dalej, że wierzchołki bezbarwne drugiego rodzaju tworzą klikę grafu G. Co więcej, każdy wierzchołek tej kliki jest sąsiedni z wierzchołkami pokolorowanymi, z których k ma różne barwy. Fakty te dają prostą metodę uzupełniania pokolorowania częściowego do pełnego w czasie nie przekraczającym $O(kn)$:

```
l: = 0;
for i: = 1, 2, ..., n do
  if wierzchołek v_i jest bezbarwny I rodzaju then
    begin
      znajdź najmniejsze j takie, że V_j ∪ {v_i} jest zbiorem niezależnym;
      V_j ← V_j ∪ {v_i}
    end v_i – I rodzaju
  else if wierzchołek v_i jest bazbarwny II rodzaju then
    begin
      l: = l+1;
      V_{k+l} ← {v_i}
    end v_i – II rodzaju
```

Przeanalizujmy obecnie złożoność obliczeniową algorytmu kolorowania par. Algorytm CP składa się z trzech etapów. W pierwszym etapie tworzymy macierz podobieństw S. Wymaga to nadania $2m+n$ elementom macierzy wartości -1 oraz obliczenia wartości pozostałych elementów nieujemnych s_{ij}. Przy okazji można obliczyć stopnie wierzchołków, które będą potrzebne w dalszych etapach algorytmu. Jeżeli $S[1:n, 1:n]$ i $\mathrm{Deg}[1:n]$ są tablicami zerowymi, to odpowiedni fragment programu można przedstawić następująco:

```
for i: = 1, 2, ..., n do
  begin
    s_ii: = –1;
    for każdego v_j sąsiedniego z v_i do
```

```
        begin
          s_ij : = -1;
          Deg_i : = Deg_i + 1
        end
      end i;
    for i: = 1, 2, ..., n do
      for każdego v_j sąsiedniego z v_i do
        for każdego v_k sąsiedniego z v_i do
          if s_jk ≠ -1 then s_jk : = s_jk + 1;
```

Łatwo zauważyć, że obliczenie stopni wszystkich wierzchołków wymaga $2m$ dodawań, a wyznaczenie ujemnych elementów macierzy podobieństw $2m+n$ podstawień. Natomiast wyznaczenie nieujemnych elementów macierzy **S** wymaga

$$\sum_{i=1}^{n} \deg^2(v_i) \leqslant \sum_{i=1}^{n} (n-1)^2 = n(n-1)^2$$

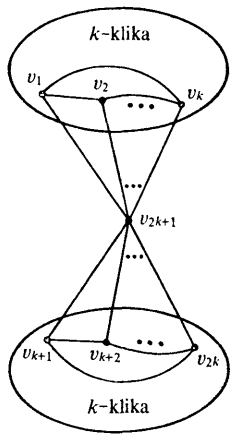
podstawień oraz

$$2 \sum_{\{i,j\} \in \bar{E}} s_{ij} < n^3$$

dodawań, gdzie $\bar{E}$ jest zbiorem krawędzi grafu $\bar{G} = (V, V \times V - E)$ dopełniającego graf G. Zatem złożoność tego etapu algorytmu wynosi $O(n^3)$. Na marginesie powyższych rozważań zauważmy, że macierz **S** można również utworzyć w oparciu o obliczenie kwadratu macierzy sąsiedztwa. Postępowanie takie może dać lepsze oszacowanie teoretyczne złożoności, gdyż mnożenie macierzy wymaga n^3 mnożeń i n^3-n^2 dodawań przy zastosowaniu algorytmu klasycznego, lecz liczbę działań można zredukować do $O(n^{2.81})$ operacji podstawowych, jeśli zastosujemy metodę Strassena [17]. Mimo to podejście takie jest praktycznie mało efektywne zwłaszcza przy małych wartościach n. W drugim etapie algorytmu CP porządkuje się wszystkie pary według ich podobieństw. Ponieważ znane są klucze, według których mamy prowadzić porządkowanie, więc sortowanie takie można zrealizować w czasie proporcjonalnym do liczby par. Zatem złożoność tego etapu jest rzędu n^2.

Właściwe kolorowanie wierzchołków następuje w trzecim etapie działania algorytmu. W każdym kroku sprawdza się czy dana para jest nieujemna w sensie podobieństwa, a następnie — jeśli tak — bada się, który wierzchołek w parze jest już pomalowany. W zależności od wyniku obu tych porównań algorytm rozgałęzia się do kroku 1, 2 lub 3. Zastanówmy się jaka jest złożoność obliczeniowa kroków 2 i 3. W drugim kroku algorytm wymaga jednego porównania i ewentualnie najwyżej $\deg(v_j)$ porównań dla stwierdzenia, czy któryś z sąsiadów wierzchołka v_j ma kolor r. Zatem w najgorszym przypadku krok ten wymaga $\deg(v_j)+1$ porównań. Natomiast dla zrealizowania działań przewidzianych w kroku 3 trzeba wykonać jedno lub dwa porównania plus ewentualnie $\deg(v_i)+\deg(v_j)$ podstawień dla zastrzeżenia kolorów sąsiadów v_i i v_j oraz najwyżej $\max\{\deg(v_i), \deg(v_j)\}$ porównań dla znalezienia dla nich odpowiedniego koloru. Zatem, gdyby przy każdym wyjściu z kroków 2 i 3

co najmniej jeden z wierzchołków bezbarwnych miał nadany jakiś kolor, to złożoność trzeciego etapu byłaby $O(m+n)$. Lecz na złożoność obliczeniową wpływają również bezskuteczne wykonania kroków 2 i 3 zwłaszcza zaś podpunktu 2(b), gdyż wymaga on najwięcej, bo $O(\deg(v_j))$ porównań. Niech $\hat{b}$ oznacza liczbę takich bezskutecznych wykonań 2(b) w trakcie działania algorytmu. Pokażemy, że istnieją grafy, dla których $\Omega(n^2)$ jest dolnym oszacowaniem liczby $\hat{b}$. Rozpatrzmy graf H_k pokazany na rysunku 2.



Rys. 2. Graf H_k

Zauważmy, że macierz podobieństw tego grafu spełnia warunek $s_{ij} \in \{-1, +1\}$ dla $i, j = 1, 2, \ldots, 2k+1$. Jeśli dodatnie pary analizowane są zgodnie z kolejnością leksykograficzną elementów macierzy S, to algorytm CP pomaluje każdą parę (v_l, v_{k+l}) kolorem l, $l = 1, \ldots, k$, lecz pomiędzy dwoma kolejnymi kolorowaniami wykona się k-krotnie podpunkt 2(b). Wobec tego $\hat{b} = k(k-1) = (n-1)(n-3)/4$. A więc trzeci etap algorytmu jest realizowany kosztem $O(n^3)$ operacji dla tych grafów. Dokładniejsza analiza złożoności w przypadku bezskutecznych wykonań podpunktu 2(b) pokazuje, że dowolny wierzchołek v_i może być pominięty najwyżej $n-3$ razy. Wobec tego sumaryczna liczba porównań nie przewyższa

$$(n-3)\sum_{v_i \in P}(\deg(v_i)+1) < (n-3)\sum_{v_i \in V}(\deg(v_i)+1) = (n-3)(2m+n) < 2(n^2+mn),$$

gdzie P jest zbiorem wierzchołków pomijanych w 2(b) w trakcie działania algorytmu. Do powyższego należy dodać $O(m+n)$ porównań potrzebnych do znalezienia odpowiednich kolorów dla wierzchołków pomalowanych. Zatem trzeci etap algorytmu CP wymaga $O(n^2+mn)$ operacji podstawowych. Tym samym ostateczna złożoność obliczeniowa całego algorytmu kolorowania par wynosi $O(n^3)$, lecz może być zmniejszona do $O(n^{2.81}+mn)$ operacji podstawowych, o ile w pierwszym etapie zastosujemy mnożenie macierzy metodą Strassena.

Metoda Wooda koloruje optymalnie grafy dwudzielne, lecz już w przypadku

$\chi(G) = 3$ jej rozwiązania mogą się różnić o dowolnie wiele od optymalnych. Johnson znalazł grafy 3-chromatyczne G_k o $n \leqslant 10k$ wierzchołkach, dla których $\mathrm{CP}(G_k) \geqslant k$. A więc podobnie jak w przypadku poprzednich algorytmów $f_{\mathrm{CP}}(n) = \Omega(n)$.

Algorytm CP daje na ogół rezultaty znacznie gorsze niż metody sekwencyjne z wyjątkiem grafów dużych i gęstych ($n \geqslant 100$, $d \geqslant 0.75$), dla których jego wyniki są lepsze niż metod LF i SL, choć gorsze od wyników LFI i SLI [11].

7. Uproszczony algorytm maksymalnych zbiorów niezależnych. Wszystkie analizowane dotąd metody mają liniową, czyli najgorszą możliwą funkcję dobroci. Dla każdego algorytmu A istnieją bowiem grafy G_k takie, że stosunek $A(G_k)/\chi(G_k)$ jest wprost proporcjonalny do liczby wierzchołków tych grafów. Nasuwa się pytanie: czy w ogóle istnieje algorytm wielomianowy, dla którego stosunek $A(G)/\chi(G)$ rośnie wolniej niż liniowo dla najgorszego układu danych? Odpowiedź jest pozytywna. W niniejszym punkcie podamy za Johnsonem [8] *uproszczony algorytm maksymalnych zbiorów niezależnych*, nazywany w skrócie AMIS (w oryginale: approximately maximum independent set), który jest najlepszym znanym algorytmem wielomianowym w tym sensie, że daje asymptotycznie najlepsze pokolorowania w najgorszym przypadku danych. Jednakże zanim to uczynimy opiszemy *przybliżony algorytm maksymalnych zbiorów niezależnych*, MIS (maximum independent set), który jest pierwowzorem algorytmu AMIS. Przybliżony algorytm maksymalnych zbiorów niezależnych określony jest następująco:

1. $i := 1$.

2. Niech U będzie zbiorem wierzchołków jeszcze nie pomalowanych. Jeśli $U = \emptyset$, to stop.

3. Niech U_i będzie najliczniejszym zbiorem niezależnym podgrafu $G|U$. Dla każdego $v \in U_i$ połóż $h(v) = i$. Następnie $i := i+1$ i wróć do kroku 2.

Johnson pokazał, że funkcja dobroci tego algorytmu $f_{\mathrm{MIS}}(n) = O(\log n)$. Jednakże algorytm MIS ma istotną wadę. Problem znalezienia najliczniejszego zbioru niezależnego należy bowiem do rodziny problemów NP-zupełnych, co oznacza, że są nikłe szanse na skonstruowanie wielomianowego algorytmu jego rozwiązywania. Te trudności sugerują uproszczenie metody polegające na zrezygnowaniu z zasady wyznaczania bezwzględnie najliczniejszego zbioru niezależnego na korzyść efektywnego wyznaczania najliczniejszego w przybliżeniu zbioru niezależnego.

Prosta metoda wyznaczania zbioru niezależnego może być zdefiniowana następująco. Niech $W_1 = V$. Jako pierwszy element — w_1 bierzemy wierzchołek o najmniejszym stopniu w grafie G. Następnie dla każdego $i = 1, 2, \ldots$, jako w_{i+1} wybieramy wierzchołek o najmniejszym stopniu w podgrafie $G|W_{i+1}$, gdzie $W_{i+1} = W_i - (\{w_i\} \cup \{v_j \in W_i : \{w_i, v_j\} \in E\})$, dopóki $W_{i+1} \neq \emptyset$. Jeżeli stopnie wierzchołków są obliczone i zapamiętane w tablicy Deg $[1:n]$, to algorytm AMIS można naszkicować jak następuje:

$U \leftarrow V$;
for $i := 1, i+1$ **while** $U \neq \emptyset$ **do**

```
begin
    V_i ← Ø; W ← U; Deg̅ ← Deg;
    comment Deg̅ jest tablicą roboczą o wymiarze zgodnym z wymiarem tabli·y Deg;
    for d: = n while W ≠ Ø do
      begin
      A: for każdego v ∈ W do if Deg̅[v] < d then
          begin
            w: = v;
            d: = Deg̅[v]
          end;
          comment w jest wierzchołkiem o aktualnie najmniejszym stopniu;
          V_i ← V_i ∪ {w}; U ← U− {w}; W ← W− {w};
      B: for każdego v_j sąsiedniego z w do
          if v_j ∈ U then
            begin
              Deg[v_j]: = Deg[v_j]− 1;
              if v_j ∈ W then
                begin
                  W ← W− {v_j};
                C: for każdego v_k sąsiedniego z v_j do
                      if v_k ∈ W then Deg̅[v_k]: = Deg̅[v_k]− 1;
                end v_j ∈ W
          end B
      end d
end i
```

W wielokrotnie cytowanej pracy [8] Johnson udowodnił, że funkcja dobroci uproszczonego algorytmu maksymalnych zbiorów niezależnych spełnia warunek

$$\Omega(n/\log n) = f_{\mathrm{AMIS}}(n) = O(n/\log n). \tag{10}$$

Zastanówmy się jaka jest złożoność obliczeniowa metody AMIS. Pętla A wykonywana jest dla każdego wierzchołka jeden raz. W kolejnym, powiedzmy k-tym, kroku zbiór W może zawierać najwyżej $n-k+1$ elementów, a więc liczba porównań potrzebnych do znalezienia wierzchołka o minimalnym stopniu nie przekracza $n-k+1$. Zatem łącznie mamy najwyżej $n(n+1)/2$ porównań. Oszacujmy teraz liczbę porównań potrzebnych do zredukowania stopni wierzchołków sąsiednich z wierzchołkami usuniętymi ze zbioru W. Pętla B wykonywana jest dla każdego wierzchołka jeden raz. Wobec tego pętla C realizowana jest najwyżej $\sum_{i=1}^{n} \deg(v_i) = 2m$ razy. Ponieważ jednorazowe wykonanie pętli C wymaga najwyżej $n-1$ porównań, więc łączna liczba tych operacji jest mniejsza od $2mn$. Zatem algorytm AMIS może być wykonany w czasie $O(n^2+mn)$.

8. Zakończenie. Dobre bądź złe zachowanie się algorytmu dla najgorszego układu danych nie rozstrzyga jeszcze o jego przydatności w praktyce. Bardzo złe działanie algorytmu w najgorszym przypadku danych ostrzega nas jedynie przed możliwym fiaskiem poszukiwań dobrego pokolorowania nie mówiąc nic o prawdopodobieństwie jego wystąpienia. Z praktyki zaś wiadomo, że najgorszy przypadek danych pojawia się niezmiernie rzadko. Z drugiej strony może zdarzyć się i tak, że najlepsze wyniki da metoda nie będąca optymalną w żadnym sensie. Zatem ważny praktycznie jest nie tyle najgorszy przypadek co „średni przypadek" danych dla danego algorytmu. To drugie podejście jest trudniejsze, gdyż wymaga określenia rozkładu danych wejściowych, a najczęściej bywa tak, że rozkłady odpowiadające rzeczywistym problemom są matematycznie niezbadane. Dotychczas dokonano jedynie analizy teoretycznej najprostszego algorytmu kolorowania, tj. algorytmu sekwencyjnego z przypadkowym wyborem wierzchołków, dla grafów losowych jednakowo prawdopodobnych o ustalonej gęstości d (patrz [10]).

Trudności związane z analizą bardziej skomplikowanych przypadków powodują, że często uciekamy się do badań empirycznych realizowanych za pomocą maszyn cyfrowych. Typowym przykładem tego nurtu jest wspomniana już praca Kubalego i Dąbrowskiego [11]. Zawarto w niej wyniki eksperymentów maszynowych związanych z następującymi algorytmami: LF, SL, LFI, SLI i CP. Algorytmy te analizowano i porównywano pod kątem liczby użytych kolorów, czasu obliczeń oraz wykorzystania pamięci maszyny. Materiałem doświadczalnym były wyniki testowania prawie tysiąca grafów pseudolosowych rozmaitych rzędów i gęstości. Grafy zapisane były w postaci macierzy sąsiedztwa **A**. Dla grafu rzędu n i zakładanej gęstości d macierz sąsiedztwa otrzymywana była w następujący sposób:

$$a_{ij} = a_{ji} = \begin{cases} 1, & \text{gdy } i \neq j \wedge 0 \leqslant r_k \leqslant d, \\ 0 & \text{w przypadku przeciwnym,} \end{cases}$$

gdzie $r_1, r_2, \ldots, r_k, \ldots$ jest ciągiem liczb pseudolosowych o rozkładzie równomiernym z przedziału (0, 1). Do generacji liczb pseudolosowych użyto kongruencyjnego generatora multyplikatywnego. Tak otrzymane macierze **A** umieszczane były na bitach słów maszynowych. Wymagana pamięć komputera wynosi wówczas $n\lceil n/w \rceil$ komórek, gdzie w jest liczbą bitów w komórce (w przypadku maszyny Odra 1204, na której przeprowadzono eksperymenty $w = 24$). Rozwiązanie to obok niezaprzeczalnych wad ma tę zaletę, że umożliwia kolorowanie grafów dużych i gęstych. Z punktu widzenia szybkości kolorowania najlepszym okazał się algorytm LF. Kolejność pozostałych metod była następująca: LFI, SL, SLI, CP. Biorąc za podstawę oceny ilość użytych barw metody sekwencyjne z wymienianiem kolorów wyraźnie górowały nad pozostałymi we wszystkich przedziałach gęstości grafów. I tak, dla grafów rzadkich, to znaczy o gęstości $d \leqslant 0.125$, na pierwszym miejscu *ex aequo* mieliśmy LFI i SLI, dalej na równi LF i SL, a na końcu metodę CP. Dla grafów średnio gęstych, tj. $0.25 \leqslant d \leqslant 0.5$, ponownie najlepsze były obie metody sekwencyjne z wymienianiem, a dalej LF, SL i CP. W przypadku grafów gęstych, mianowicie o $d \geqslant 0.75$, na trzecią pozycję za LFI i SLI przesunęła się metoda kolorowania par,

następnie mieliśmy LF i na końcu SL. Niekiedy w zastosowaniach ważna jest oszczędność pamięci maszyny cyfrowej. Pod względem zapotrzebowania na pamięć zdecydowanie najgorszą okazała się metoda CP, natomiast kolejność pierwszych czterech metod była następująca: LF, SL, LFI, SLI. Reasumując, wydaje się, że przyszłemu użytkownikowi można polecić którąś z metod sekwencyjnych z wymienianiem kolorów, np. LFI. Metodę tę warto by jeszcze skonfrontować z asymptotycznie najlepszym algorytmem AMIS.

Podziękowanie. Autor wyraża podziękowanie drowi Maciejowi M. Syśle za cenne uwagi krytyczne, które zostały uwzględnione w redakcji tej pracy.

Prace cytowane

[1] K. Appel, W. Haken, *Every planar map is four colorable*, I, Illinois J. Math. 21.3 (1977), str. 429–490.

[2] K. Appel, W. Haken, J. Koch, *Every planar map is four colorable*, II, ibidem 21.3 (1977), str. 491–567.

[3] J. R. Brown, *Chromatic scheduling and the chromatic number problem*, Manag. Sci. 19.4 (1972), str. 456–463.

[4] H. Burlaga, *Suboptymalne metody kolorowania grafów*, Biuletyn WAT 22.2 (1973), str. 63–69.

[5] N. Christofides, *An algorithm for the chromatic number of a graph*, Computer J. 14.1 (1971), str. 38–39.

[6] M. R. Garey, D. S. Johnson, *The complexity of near-optimal graph coloring*, Journal of ACM 23.1 (1976), str. 43–49.

[7] —, —, *„Strong" NP-completeness results: Motivation, examples and implications*, ibidem 25.3 (1978), str. 499–508.

[8] D. S. Johnson, *Worst case behavior of graph coloring algorithms*, w: *Proc. 5-th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, Utilitas Mathematica Publishing (1974), str. 513–527.

[9] R. M. Karp, *Reducibility among combinatorial problems*, w: *Complexity of Computer Computations*, red. R. E. Miller i J. W. Thather, Plenum Press (1972), str. 85–104.

[10] —, *The probabilistic analysis of some combinatorial search algorithms*, w: *Algorithms and Complexity. New Directions and Recent Results*, red. J. E. Traub, Academic Press (1976), str. 1–19.

[11] M. Kubale, J. Dąbrowski, *Empirical comparison of efficiency of some graph colouring algorithms*, Arch. Autom. i Telemech. 23. 1–2 (1978), str. 129–139.

[12] E. L. Lawler, *A note on the complexity of the chromatic number problem*, Inform. Process. Letters 5.3 (1976), str. 66–67.

[13] R. J. Lipton, R. E. Miller, *A batching method for coloring planar graphs*, ibidem 7.4 (1978), str. 185–188.

[14] D. W. Matula, G. Marble, D. Isaacson, *Graph coloring algorithms*, w: *Graph Theory and Computing*, red. R. Read, Academic Press (1972), str. 109–122.

[15] R. L. Rivest, J. Vuillemin, *On recognizing graph properties from adjacency matrices*, Theoret. Comp. Sci. 3.3 (1976), str. 371–384.

[16] S. I. Roschke, A. L. Furtado, *An algorithm for obtaining the chromatic number and an optimal coloring of a graph*, Inform. Process. Letters 2. 2 (1973), str. 34–38.

[17] V. Strassen, *Gaussian elimination is not optimal*, Numer. Math. 13. 4 (1969), str. 354–356.

[18] M. M. Sysło, *O złożoności obliczeniowej problemów kombinatoryki i teorii grafów*, Matematyka Stosowana 16 (1980), str. 53–98.

[19] S. Tsukiyama, H. Ozaki, H. Ariyoshi, *An algorithm for generating all the maximal independent sets*, Electronics and Comm. in Japan, 59-A.6 (1976), str. 1–8.

[20] C. C. Wang, *An algorithm for the chromatic number problem*, Journ. ACM 21.3 (1974), str. 385–391.

[21] J. Weissman, *Boolean algebra, map coloring, and interconnections*, Amer. Math. Monthly 69.7 (1972), str. 608–613.

[22] D. J. Welsh, M. B. Powell, *An upper bound for the chromatic number of a graph and its application to timetabling problems*, Computer J. 10. 1 (1967), str. 85–86.

[23] D. C. Wood, *A technique for coloring a graph applicable to large scale timetabling problems*, ibidem 12.4 (1969), str. 317–319.

Dodane w korekcie. Od momentu napisania tego artykułu w roku 1978 mamy do czynienia z istnym zalewem prac poświęconych nowym, bardziej wydajnym metodom obliczeniowym. Część z nich może być wykorzystana do usprawnienia algorytmów kolorowania grafów.

Przede wszystkim nie jest już aktualna informacja, iż metoda partiowa [13] o złożoności $O(n \log n)$ jest najlepszym sposobem kolorowania grafów płaskich. Obecnie najbardziej skutecznym sposobem kolorowania takich grafów jest algorytm liniowy [24], który gwarantuje pokolorowanie grafu płaskiego co najwyżej 5 kolorami. Poza tym warto odnotować postęp w zakresie szybkiego mnożenia macierzy. Wyróżnić tutaj można prace Włochów [25], [26] dotyczące algorytmów przybliżonych, tzw. APA (arbitrary precision approximating) oraz publikacje Pana [27], [28] poświęcone algorytmom dokładnym EC (exactly computing). Ponadto trzeba wymienić $O(n^2)$ algorytm Santoro [29] mnożenia macierzy rozrzedzonych. Ten ostatni zastosowany w algorytmie CP do obliczenia macierzy podobieństw pozwala na obniżenie jego złożoności obliczeniowej do poziomu $O(n^2)$, o ile kolorowany graf jest rzadki (tj. gdy $m = O(n)$).

[24] N. Chiba, T. Nishizeki, N. Saito, *A linear 5-coloring algorithm of planar graphs*, w: *Tohoku Univ., Tsuken Symp. on Graph Theory and Algorithms* (1980), str. 11–23.

[25] D. Bini, M. Capovani, G. Lotti, F. Romani, *$O(n^{2.7799})$ complexity for $n \times n$ approximate matrix multiplication*, Inform. Process. Letters 8.5 (1979), str. 234–235.

[26] D. Bini, *Relations between exact and approximate bilinear algorithms. Applications*, Calcolo 17.1 (1980), str. 87–97.

[27] V. Ya. Pan, *New fast algorithms for matrix operations*, SIAM J. Comp. 9.2 (1980), str. 321–342.

[28] V. Ya. Pan, *New combinations of methods for the acceleration of matrix multiplication*, Comp. Math. with Appl. 7.1 (1981), str. 73–125.

[29] N. Santoro, *Four $O(n^2)$ multiplication methods for sparse and dense Boolean matrices*, Proc. 10-th Conf. on Numerical Mathematics and Computing, Winnipeg, Manitoba (1980).