Hanna FURMAŃCZYK, Artur KOLIŃSKI
Uniwersytet Gdański

WYDAJNY ALGORYTM DLA r-SPRAWIEDLIWEGO KOLOROWANIA GRAFÓW

Streszczenie. W pracy rozważamy problem r-sprawiedliwego właściwego kolorowania grafów, w którym rozmiar klas kolorów może różnić się maksymalnie o wartość r. Taki model ma szerokie zastosowania praktyczne.
Prezentujemy wyniki dotyczące złożoności problemu dla wybranych klas grafów. Podajemy przykład klasy grafów, dla której klasyczne kolorowanie jest trudne obliczeniowo, kolorowanie sprawiedliwe ($r = 1$) jest rozwiązywalne w czasie wielomianowym, podczas gdy kolorowanie r-sprawiedliwe z $r \geqslant 2$ jest problemem NP-trudnym. Podajemy również bardzo wydajny algorytm dla r-sprawiedliwego kolorowania grafów, który został przetestowany na grafach pochodzących z biblioteki benchmarków DIMACS. Część wyników została zaprezentowana w niniejszej pracy.

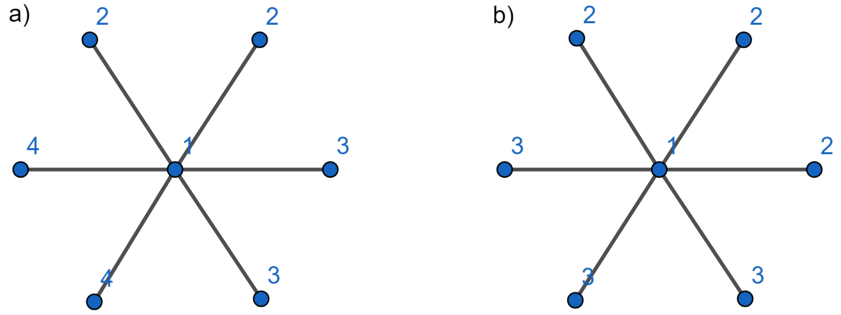
EFFICIENT ALGORITHM FOR r-EQUITABLE GRAPH COLORING

Summary. In the paper we consider the problem of r-equitable proper colouring of graphs in which the size of colour classes may differ by at most r. Such a graph colouring model has many applications.
We present computational complexity results for some graph classes. We give also an example of a graph class that proper colouring is hard, an equitable colouring ($r = 1$) can be achieved in polynomial time, while an r-equitable colouring with $r \geqslant 2$ is again NP-hard. We formulate also very efficient algorithm for r-equitable colouring that has been tested on DIMACS graphs. Part of them are presented hereby.

1. Wprowadzenie i podstawowe definicje

W niektórych dyskretnych procesach przemysłowych mamy do czynienia z problemem podziału zbioru zawierającego konflikty na podzbiory bezkonfliktowe. Bardzo często chcemy, aby podział ten był zbalansowany - aby podzbiory, na które dokonaliśmy podziału były takiej samej mocy lub, aby różnica pomiędzy najbardziej a najmniej licznym podzbiorem nie była za duża. Takie sytuacje mogą być modelowane za pomocą, odpowiednio, sprawiedliwego i r-sprawiedliwego kolorowania grafu konfliktu.

Definicja 1. *Niech G będzie grafem prostym. Mówimy, że G jest sprawiedliwie k-kolorowalny, jeżeli jego zbiór wierzchołków V można podzielić na k zbiorów nieza-*



Rys. 1. Optymalne pokolorowanie gwiazdy $K_{1,6}$: a) sprawiedliwie, b) 2-sprawiedliwie.

leżnych (klas kolorów) $V_1, \ldots, V_k$ takich, że $||V_i| - |V_j|| \leqslant 1$ dla wszystkich $i \neq j$. Najmniejszą liczbę k, dla której istnieje sprawiedliwe k-pokolorowanie nazywamy sprawiedliwą liczbą chromatyczną i oznaczamy przez $\chi_=(G)$.

Definicja 2. *Niech G będzie grafem prostym i niech $r \geqslant 0$. Mówimy, że G jest r-sprawiedliwie k-kolorowalny, jeżeli jego zbiór wierzchołków V można podzielić na k zbiorów niezależnych (klas kolorów) $V_1, \ldots, V_k$ takich, że $||V_i| - |V_j|| \leqslant r$ dla wszystkich $i \neq j$. Najmniejszą liczbę k, dla której istnieje r-sprawiedliwe k-pokolorowanie nazywamy r-sprawiedliwą liczbą chromatyczną i oznaczamy przez $\chi_{r=}(G)$.*

Oczywiście, prawdziwa jest równość: $\chi_=(G) = \chi_{1=}(G)$.

Problem sprawiedliwego kolorowania grafów został wprowadzony w 1973 przez Meyera [12], natomiast jego uogólnienia do r-sprawiedliwego kolorowania grafów dokonali w 2011 roku Hertz i Ries [8]. Meyer w swojej pracy jako przykład zastosowania wprowadzanego modelu podał problem oczyszczania miasta. Wierzchołki w jego grafie reprezentowały trasy wywozu śmieci, zaś dwa wierzchołki były połączone krawędzią, gdy odpowiadające im trasy nie mogły być obsłużone tego samego dnia. Problem przydziału jednego z sześciu dni pracy do każdej trasy sprowadza się do pokolorowania grafu sześcioma kolorami. W praktyce, ze względu na ograniczenia taboru chcemy, aby każdego dnia obsłużyć możliwie taką samą liczbę tras, a zatem nasz graf należy pokolorować sprawiedliwie sześcioma kolorami. W niektórych przypadkach możemy dopuszczać trochę mniej zbalansowany podział tras, np. gdy dopuszczalna różnica obsługiwanych tras pomiędzy dowolnymi dwoma dniami pracy wynosi 2, nasz graf należy pokolorować 2-sprawiedliwie sześcioma kolorami. Oba modele kolorowania grafów mają szerokie zastosowanie również w szeregowaniu zadań (por. np. [5]).

2. Złożoność obliczeniowa problemów. Sprawiedliwe vs r-sprawiedliwe kolorowanie grafów

W ogólności, problem optymalnego sprawiedliwego kolorowania wierzchołków grafów jest problemem trudnym obliczeniowo. Dla przykładu mamy

Twierdzenie 1. ([3]) *Problem odpowiedzi na pytanie: czy $\chi_=(G) \leqslant 3$ jest NP-zupełny również wtedy, gdy G jest grafem krawędziowym grafu kubicznego.*

Oczywiście, problem kolorowania r-sprawiedliwego, jako szerszy - z dowolną wartością r, również w ogólności jest trudny. Poniżej przedstawiamy znane dotychczas klasy grafów, dla których oba rozważane problemy są trudne:

- grafy krawędziowe grafów kubicznych [3],
- grafy dwudzielne [2],
- kografy [2], a zatem również grafy permutacyjne i doskonałe,
- grafy przedziałowe [2], a zatem również grafy cięciwowe,
- korony grafów kubicznych [6].

Zauważmy, że istnieją klasy grafów, np. grafy dwudzielne, czy korony grafów kubicznych, dla których problem kolorowania klasycznego jest łatwy, a problem kolorowania sprawiedliwego, czy r-sprawiedliwego jest trudny obliczeniowo. Nasuwa się pytanie, czy znane są grafy, dla których problem kolorowania sprawiedliwego jest łatwy, a problem kolorowania r-sprawiedliwego z $r \geqslant 2$ - trudny. Odpowiedź jest pozytywna. Co więcej jest to klasa grafów, dla których problem kolorowania klasycznego jest trudny! Niech G będzie dowolnym grafem, dla którego problem kolorowania klasycznego jest NP-trudny, np. grafem bez pazura ($K_{1,3}$) [11]. Rozważmy graf $G' = G + N_1$, czyli taki, dla którego $V(G') = V(G) \cup \{v\}$ oraz $E(G') = E(G) \cup \{uv : u \in V(G)\}$. Poniżej rozważamy trzy modele kolorowania dla grafu G'.

Kolorowanie klasyczne Jest oczywiste, że problem optymalnego pokolorowania grafu $G' = G + N_1$ jest problemem trudnym obliczeniowo.

Kolorowanie sprawiedliwe Problem optymalnego sprawiedliwego pokolorowania grafu $G' = G + N_1$ jest rozwiązywalny w czasie wielomianowym. Zauważmy, że kolor przydzielony wierzchołkowi v nie może być użyty ponownie w grafie G'. Ponieważ pokolorowanie ma być sprawiedliwe, każdego innego koloru możemy użyć maksymalnie dwukrotnie. Oznacza to, że zbiór wierzchołków grafu G należy podzielić na możliwie najmniejszą liczbę klas kolorów rozmiaru maksymalnie 2, co sprowadza się do znalezienia maksymalnego skojarzenia w dopełnieniu grafu G, co z kolei może być wykonane w czasie $O(\sqrt{|V(G)||E(\overline{G})|})$ [13].

Kolorowanie r-sprawiedliwe Problem optymalnego r-sprawiedliwego pokolorowania grafu $G' = G + N_1$ dla $r \geqslant 2$ jest NP-trudny. Zauważmy, że kolor przydzielony wierzchołkowi v nie może być użyty ponownie w grafie G'. Ponieważ pokolorowanie ma być r-sprawiedliwe, każdego innego koloru możemy użyć maksymalnie $r + 1$ razy. Oznacza to, że graf G musi być pokolorowany w taki sposób, aby rozmiar żadnej z klas kolorów nie przekraczał $r + 1$, czyli w sposób m-ograniczony z $m = r + 1$. Wiemy, że problem m-ograniczonego, $m \geqslant 3$, kolorowania grafów dowolnych jest problemem trudnym [7].

Podsumowując temat złożoności, nieznajomość wielomianowego rozwiązania problemu sprawiedliwego, czy r-sprawiedliwego kolorowania grafów w przypadku ogólnym oznacza, że w praktyce zmuszeni jesteśmy poszukiwać algorytmów przybliżonych, czy heurystyk.

3. Algorytmy

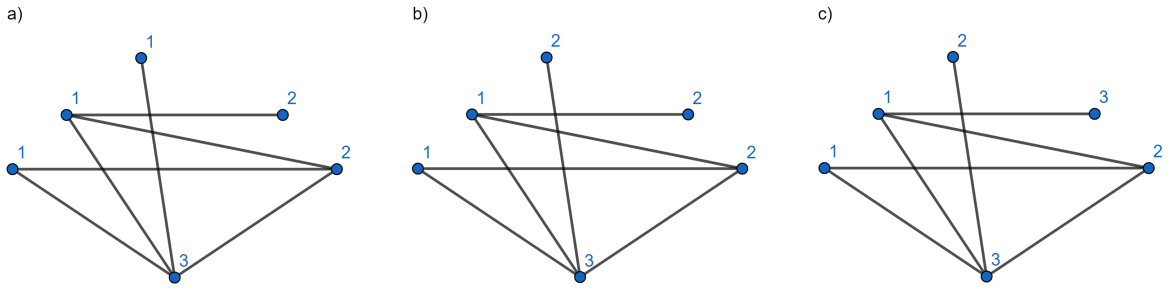
Furmańczyk, Jastrzębski i Kubale [4] podali dwa algorytmy sprawiedliwego kolorowania grafów bazujące na heurystykach zwracających pokolorowanie właściwe, niekoniecznie sprawiedliwe. Na potrzeby porównania z naszym algorytmem, który omówimy w dalszej części podrozdziału, będziemy bazować na heurystykach LF oraz SLF [10]. Algorytmy z [4] dokonywały transformacji pokolorowania właściwego do pokolorowania sprawiedliwego. Doświadczenia komputerowe opisane w pracy [4] wykazały wyższość jednego z nich, nazwanego algorytmem FJK.

```
algorytm FJK(G):
begin
    przelicz moce poszczególnych klas kolorów;
    while kolorowanie nie jest sprawiedliwe do begin
        znajdź wierzchołki pomalowane najbardziej
            liczną klasą kolorów;
        if jeżeli można zamienić wierzchołek najbardziej licznej
            klasy koloru z najmniej liczną klasą koloru
        then zrób to else wprowadź nowy kolor
            dla wierzchołka z najbardziej licznej klasy koloru;
    end;
end;
```

Prosta transformacja warunku w pętli **while** pozwala zaadaptować powyższy algorytm tak, aby zwracał pokolorowanie r-sprawiedliwe, które być może wykona mniej iteracji pętli **while** i pożądane pokolorowanie zwróci szybciej. Okazuje się, że dokonanie głębszej transformacji algorytmu FJK daje nadspodziewanie dobre rezultaty. K-FJK jest to algorytm bazujący na FJK zmodyfikowany w taki sposób, aby mógł mniejszą liczbą kolorów pokolorować dany graf r-sprawiedliwie. Algorytm wygląda następująco:

```
algorytm K-FJK(G):
begin
    przelicz moce poszczególnych klas kolorów
    oraz posortuj je nierosnąco;
    while kolorowanie nie jest r-sprawiedliwe do begin
        znajdź wierzchołki pomalowane najbardziej
            liczną klasą kolorów;
        if jeżeli można zamienić wierzchołek najbardziej licznej
            klasy koloru z najmniej liczną klasą koloru
        then zrób to
        else if można zamienić wierzchołek najbardziej licznej
            klasy koloru na każdy inny z mniejszą liczbą
            wystąpień począwszy od najmniej licznych
                then zrób to
                else wprowadź nowy kolor dla wierzchołka
                z najbardziej licznej klasy koloru;
    end;
end;
```

Oryginalny algorytm FJK jest w stanie zamieniać tylko wierzchołki pokolorowane najbardziej licznym kolorem na kolory pochodzące z wierzchołków posiadających najmniej liczne kolory. W przypadku, gdy taka zamiana nie jest możliwa, wprowadza on nowy kolor. Tutaj z pomocą przychodzi ulepszona wersja algorytmu - algorytm K-FJK, która w przypadku niemożności takiej zamiany nie wprowadza tak szybko nowego koloru. Zamiast tego algorytm próbuje zamienić kolor wierzchołka z najbardziej licznym kolorem na kolory pochodzące ze wszystkich mniej licznych klas kolorów począwszy od najmniej licznych kolorów (por. rys. 2). Takie rozwiązanie wprowadza nowy kolor tylko w przypadku niemożności zamiany na żaden inny mniej liczny kolor. Dzięki takiemu podejściu zachodzi znacznie mniej operacji dodania nowego koloru, ponieważ algorytm dość szybko balansuje wszystkie kolory.



Rys. 2. Przykład grafu a) pokolorowanego klasycznie. b) i c) Usprawiedliwienie pokolorowania algorytmem K-FJK.

Złożoność obliczeniowa algorytmu K-FJK wynosi $O(n^3 \log n)$. Wynika ona z oszacowania liczby przekolorowań $O(n \log n)$ [4] oraz z faktu, że pojedyncze przekolorowanie zajmuje $O(n^2)$ czasu.

4. Doświadczenia komputerowe

Algorytm K-FJK został przetestowany oraz porównany z algorytmem FJK na benchmarkach z bazy DIMACS [9] (35 instancji grafów). Oba algorytmy korzystały z tych samych pokolorowań klasycznych - uzyskanych heurystykami LF oraz SLF.

Wszystkie pomiary zostały wykonane na komputerze: Intel i5-6600k 4.2GHz 16GB RAM, dysk SSD. Ze względu na dużą moc obliczeniową komputera, na którym zostały przeprowadzone doświadczenia, uzyskiwane czasy kolorowań są bardzo małe. Czas wykonania każdego algorytmu jest wyrażony w milisekundach (kolumny oznaczone jako 'ms' w poniższych tabelach). Mierzony jest jedynie czas samych algorytmów usprawiedliwiających, pomiar czasu nie uwzględnia wstępnego pokolorowania LF oraz SLF. Ze względu na ograniczenie miejsca, prezentujemy tylko część wyników.

W tabeli 1 prezentujemy liczbę kolorów użytych do pokolorowania wybranych grafów. Kolumna LF oznacza liczbę kolorów użytych przez algorytm LF (pokolorowanie klasyczne), kolumny LF_{FJK} oraz LF_{K-FJK} oznaczają liczbę kolorów użytych do pokolorowania 1-sprawiedliwego odpowiednio przez algorytmy FJK oraz K-FJK. Analogicznie prezentujemy wyniki dla algorytmów FJK oraz K-FJK bazujące na pokolorowaniu klasycznym uzyskanym za pomocą heurystyki SLF (tabele 2-4).

Wiedząc, że do obu algorytmów usprawiedliwiających (FJK oraz K-FJK) wcho-

Tabela 1
Wyniki dla 1-sprawiedliwego pokolorowania bazującego na algorytmie LF.

Nazwa grafu	liczba wierzchołków	LF	LF_{FJK}	ms	LF_{K-FJK}	ms	Zysk
C2000.5	2000	219	513	2081	292	3038	43%
dsjc250.5	250	41	55	7	49	14	11%
dsjc500.1	500	18	22	122	19	20	14%
dsjc500.5	500	71	129	48	94	90	27%
dsjc500.9	500	169	287	88	173	82	40%
dsjc1000.1	1000	29	42	84	30	104	29%
dsjc1000.9	1000	313	578	590	342	621	41%
dsjr500.5	500	134	143	19	137	16	4%
flat300_28_0	300	45	82	14	51	17	38%
flat1000_50_0	1000	123	219	233	156	489	29%
latin_square	900	213	365	344	302	892	17%
miles1000	128	43	43	0	43	0	0%
miles1500	128	73	74	1	73	0	1%
myciel4	23	5	7	0	5	0	29%
queen8_8	64	13	14	1	13	1	7%
r1000.5	1000	259	281	134	260	121	7%
zeroin.i.1	211	49	55	3	50	6	9%
zeroin.i.2	211	30	47	15	37	33	21%
zeroin.i.3	206	30	47	5	37	23	21%

Tabela 2
Wyniki dla 1-sprawiedliwego pokolorowania bazującego na algorytmie SLF.

Nazwa grafu	liczba wierzchołków	SLF	SLF_{FJK}	ms	SLF_{K-FJK}	ms	Zysk
C2000.5	2000	224	435	1360	289	1637	34%
dsjc250.5	250	41	63	6	43	3	32%
dsjc500.1	500	18	21	11	18	6	14%
dsjc500.5	500	74	134	43	89	45	34%
dsjc500.9	500	175	291	71	179	15	38%
dsjc1000.1	1000	30	42	67	31	41	26%
dsjc1000.9	1000	317	580	445	344	184	41%
dsjr500.5	500	129	138	6	132	10	4%
flat300_28_0	300	47	73	9	51	6	30%
flat1000_50_0	1000	124	265	253	146	175	45%
latin_square	900	219	474	384	304	543	36%
miles1000	128	43	44	0	43	0	2%
miles1500	128	73	73	0	73	0	0%
myciel4	23	5	5	0	5	0	0%
queen8_8	64	15	17	0	15	0	12%
r1000.5	1000	250	279	49	259	94	7%
zeroin.i.1	211	49	56	3	50	3	11%
zeroin.i.2	211	30	48	19	37	13	23%
zeroin.i.3	206	30	57	11	37	9	35%

dzi identycznie pokolorowany graf, za pomocą heurystyki LF lub SLF, jesteśmy w stanie skupić się na porównaniu samych algorytmów oraz ich czasów. W ostatniej kolumnie powyższych tabel, nazwanej 'Zysk', wyliczona została wartość określająca o ile procent algorytm K-FJK okazał się lepszy od algorytmu FJK, według wzoru:

$$\frac{\mathsf{FJK}(G) - \mathsf{K\text{-}FJK}(G)}{\mathsf{FJK}(G)},$$

gdzie $\mathsf{FJK}(G)$ oraz $\mathsf{K\text{-}FJK}(G)$ oznacza liczbę kolorów użytą do pokolorowania grafu G przez algorytm, odpowiednio, FJK, czy K-FJK. Jak nietrudno zauważyć w wielu przypadkach przewaga algorytmu K-FJK jest bardzo duża. W przypadku kilku grafów algorytm K-FJK zwrócił pokolorowanie sprawiedliwe, do którego użył o ponad 40% kolorów mniej w stosunku do algorytmu FJK (grafy: C200.5, dsjc1000.9, flat1000_50_0 w tabelach 1 oraz 2). Obok przewagi w liczbie użytych kolorów mamy również przewagę wynikającą z czasu potrzebnego na obliczenia. W bardzo wielu przypadkach lepsze

Tabela 3
Wyniki dla 2-sprawiedliwego pokolorowania bazującego na algorytmie SLF.

Nazwa grafu	liczba wierzchołków	SLF	SLF_{FJK}	ms	SLF_{K-FJK}	ms	Zysk
C2000.5	2000	224	391	1074	255	918	35%
dsjc250.5	250	41	53	3	43	3	19%
dsjc500.1	500	18	21	14	18	5	14%
dsjc500.5	500	74	94	18	76	17	19%
dsjc500.9	500	175	195	10	175	4	10%
dsjc1000.1	1000	30	39	51	30	30	23%
dsjc1000.9	1000	317	387	120	319	22	18%
dsjr500.5	500	129	129	1	129	1	0%
flat300_28_0	300	47	56	4	48	2	14%
flat1000_50_0	1000	124	223	193	146	171	35%
latin_square	900	219	397	274	235	244	41%
miles1000	128	43	43	0	43	0	0%
miles1500	128	73	73	0	73	0	0%
myciel4	23	5	5	0	5	0	0%
queen8_8	64	15	15	0	15	0	0%
r1000.5	1000	250	254	14	251	18	1%
zeroin.i.1	211	49	54	2	50	3	7%
zeroin.i.2	211	30	46	11	34	9	26%
zeroin.i.3	206	30	55	20	34	10	38%

Tabela 4
Wyniki dla 4-sprawiedliwego pokolorowania bazującego na algorytmie SLF.

Nazwa grafu	liczba wierzchołków	SLF	SLF_{FJK}	ms	SLF_{K-FJK}	ms	Zysk
C2000.5	2000	224	289	441	225	141	22%
dsjc250.5	250	41	44	0	41	0	7%
dsjc500.1	500	18	20	15	18	7	10%
dsjc500.5	500	74	76	2	74	3	3%
dsjc500.9	500	175	175	0	175	0	0%
dsjc1000.1	1000	30	38	43	30	27	21%
dsjc1000.9	1000	317	317	0	317	0	0%
dsjr500.5	500	129	129	0	129	0	0%
flat300_28_0	300	47	50	1	48	1	4%
flat1000_50_0	1000	124	147	48	124	22	16%
latin_square	900	219	243	62	219	73	10%
miles1000	128	43	43	0	43	0	0%
miles1500	128	73	73	0	73	0	0%
myciel4	23	5	5	0	5	0	0%
queen8_8	64	15	15	0	15	0	0%
r1000.5	1000	250	250	1	250	1	0%
zeroin.i.1	211	49	51	2	49	2	4%
zeroin.i.2	211	30	41	2	32	4	22%
zeroin.i.3	206	30	48	6	32	7	33%

pokolorowanie uzyskujemy w znacznie krótszym czasie. Analizując tabele 3 oraz 4, prezentujące wyniki dla r-sprawiedliwego pokolorowania grafów, $r \geqslant 2$, przewaga naszego algorytmu jest nadal duża (sięga 41% dla $r = 2$ oraz 33% dla $r = 4$). Oczywiście, wraz ze wzrostem r liczba użytych kolorów maleje.

W literaturze opisanych jest niewiele algorytmów dla sprawiedliwego kolorowania grafów, czyli dla modelu r-sprawiedliwego z $r = 1$. Nasza praca jest pierwszą, która podaje algorytmy dla $r \geqslant 2$. Ponadto, warto również podkreślić, że nasze wyniki dla $r = 1$ dają się porównać z wynikami uzyskanymi w pracy [1]. Autorzy tej pracy podali dwa algorytmy typu *branch-and-cut*. Za pomocą metody programowania całkowitoliczbowego wyznaczyli oni dolną i górną granicę na liczbę kolorów potrzebną do sprawiedliwego pokolorowania grafu. Swoje algorytmy przetestowali miedzy innymi na

grafach DIMACS, stąd możliwość porównania ich wyników z naszymi rezultatami. W wielu przypadkach rozważanych grafów ich dolne i górne oszacowania były takie same, czyli uzyskali oni pokolorowanie optymalne. Ale co najważniejsze, nasze wyniki bardzo często pokrywały się z ich rezultatami, w niewielu różniły się np. o jeden kolor. Pozwala to nam wyciągnąć wniosek, że nasze rezultaty są bardzo bliskie optymalnym wartościom sprawiedliwej liczby chromatycznej i co również ważne, czas uzyskania wyników w naszym przypadku jest znacznie krótszy niż czas algorytmów z pracy [1].

LITERATURA

1. Bahiense L., Frota Y., Noronha T.F., Ribeiro C.C.: A branch-and-cut algorithm for the equitable coloring problem using a formulation by representatives. Discrete Applied Mathematics, 164(1), 2014, p. 34–46.
2. Bodlaender H. L. , Jansen K.: Restrictions of graph partition problems - Part I. Theoretical Computer Science, 148, 1995, p. 93–109.
3. Furmańczyk H.: Equitable coloring, in: Graph Colorings (ed. M. Kubale), Contemporary Mathematics 352, AMS, Ann Arbor (2004).
4. Furmańczyk H., Jastrzębski A., Kubale M.: Equitable coloring of graphs. Recent theoretical results and new practical algorithms. Archives of Control Sciences, 26(3), 2016, p. 281–295.
5. Furmańczyk H., Kubale M.: Equitable and semi-equitable coloring of cubic graphs and its application in batch scheduling. Archives of Control Sciences, 25(1), 2015, p. 109–116.
6. Furmańczyk H., Kubale M.: Equitable coloring of corona products of cubic graphs is harder than ordinary coloring. Ars Mathematica Contemporanea, 10(2), 2016, p. 333–347.
7. Hansen P., Hertz A., Kuplinsky J.: Bounded vertex colorings of graphs. Discrete Math., 111, 1993, p. 305–312.
8. Hertz A., Ries B.: On r-equitable colorings of trees and forests. Les Cahiers du GERARD, 2011, G-2011-40.
9. Johnson D.S., Trick M.A.: Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge, in: DIMACS Series in discrete mathematics and theoretical computer science, Vol. 26, American Mathematical Society, Providence, 1996.
10. Kosowski A., Manuszewski K.: Classical coloring of graphs, in: Graph Colorings (ed. M. Kubale), Contemporary Mathematics 352, AMS, Ann Arbor (2004).
11. Kral D., Kratochvil J., Tuza Z., Woeginger G.J.: Complexity of coloring graphs without forbidden induced subgraphs. Proc. 27th International Workshop on Graph-Theoretic Concepts in Computer Science WG'01, LNCS 2204, 2001, p. 254–262.
12. Meyer, W.: Equitable coloring. Amer. Math. Monthly, 80, 1973, p. 920–922.
13. Micali S., Vazivani V.V.: An $O(\sqrt{n}|E|)$ algorithm for finding maximum matching in general graphs. Proc. 21st IEEE Annual Symposium on Foundations of Computer Science (1980), p. 17–27.