Uniwersytet im. Adama Mickiewicza
Wydział Matematyki i Informatyki

Krzysztof Krzywdziński

Rozproszone algorytmy aproksymacyjne dla grafów dyskowych

rozprawa doktorska

Promotor:
prof. dr hab. Michał Karoński

Poznań 2010

Składam serdeczne podziękowania
panu profesorowi doktorowi habilitowanemu
Michałowi Karońskiemu za jego
nieocenioną pomoc.

Dziękuję także serdecznie mojej rodzinie,
bez której wsparcia i pomocy
praca ta nie mogłaby powstać.

Spis treści

1. Wstęp

1.1. Wprowadzenie

W ciągu ostatnich kilku lat obserwuje się rozwój rozległych sieci informatycznych i telekomunikacyjnych, takich jak sieci internetowe, sieci stron WWW oraz sieci ad-hoc (na przykład sieci komórkowe i sieci sensorowe). Większość z nich ma charakter zdecentralizowany, czyli każdy ich element stanowi autonomiczną jednostkę, która ma zdolność do wykonywania lokalnych obliczeń oraz komunikacji ze swoimi bezpośrednimi sąsiadami. Sieci zdecentralizowane nie posiadają ani globalnej wiedzy o całej strukturze ani centralnej jednostki obliczeniowej. W związku z tym implementacja w nich większości klasycznych algorytmów opartych na scentralizowanym modelu wiedzy jest trudna lub niemożliwa. W celu rozwiązania problemów istotnych ze względu na zastosowania sieci zdecentralizowanych przyjmuje się zatem tak zwany rozproszony model obliczeń. Dostosowany jest on do charakterystyki sieci zdecentralizowanych, gdyż przyjmuje się w nim, że zarówno obliczenia jak i wysyłanie informacji wykonywane są jednocześnie, oraz niezależnie w wielu miejscach.

Istotnym zagadnieniem jest rozwiązanie w modelu rozproszonym problemów związanych z zastosowaniami sieci zdecentralizowanych. Są to między innymi: rozsyłanie informacji, współdzielenie zasobów, zapewnienie bezpieczeństwa itp. Część z tych problemów, nawet w modelu scentralizowanym, mieści się w klasie problemów bardzo trudnych obliczeniowo a większość z nich nie może być rozwiązana dokładnie bez globalnej wiedzy o sieci. W dodatku w praktyce zwykle istotniejsze jest to, jak szybko szukany obiekt może być uzyskany niż dokładność konstrukcji. Dlatego wystarczające są rozwiązania, które tylko przybliżają szukane struktury. Obiekty takie nazywamy aproksymacyjnymi a algorytmy je konstruujące – algorytmami aproksymacyjnymi.

Celem rozprawy jest przedstawienie szybkich algorytmów aproksymacyjnych rozwiązujących w modelu rozproszonym niektóre z istotnych ze względu na zastosowania problemów. Rozważania skupiają się na zagadnieniach związanych z sieciami zdecentralizowanymi, w których komunikacja odbywa się drogą radiową. Przykładem takich sieci są sieci sensorowe, sieci komórkowe, sieci radiowe i ogólnie – sieci ad–hoc. Ich teoretycznym modelem są jednostkowe grafy dyskowe, w których wierz-

chołki są punktami na płaszczyźnie a połączenia powstają między wierzchołkami w odległości co najwyżej 1. W modelu tym wierzchołki odpowiadają jednostkom obliczeniowym sieci a krawędzie przedstawiają możliwość komunikacji między nimi (tzn. fakt, że elementy są wzajemnie w swoim zasięgu).

Rozważania dotyczyć będą trzech istotnych klas problemów związanych z działaniem i strukturą sieci radiowych.

Pierwszą z nich są zagadnienia związane ze znalezieniem optymalnej struktury komunikacyjnej i z bezkolizyjnym rozsyłaniem informacji. Im poświęcony jest rozdział 2. W nim jako pierwszy przedstawiony jest algorytm konstruujący aproksymację minimalnego drzewa rozpinającego, czyli strukturę minimalizującą sumaryczny koszt rozesłania informacji w całej sieci. Kolejne algorytmy rozpatrywane w tym rozdziale podają rozwiązanie różnych wersji klasycznego problemu rozpowszechniania komunikatów w sieciach radiowych, w których zachodzi zjawisko interferencji.

Kolejnym z problemów poruszonych w niniejszej rozprawie jest konstrukcja podziału grafu na klastry (fragmenty). Podziały takie wykorzystywane są do rozwiązywania w sposób rozproszony wielu innych problemów związanych z sieciami zdecentralizowanymi. W rozdziale 3 zaprezentowane są dwa algorytmy rozproszone dzielące jednostkowe grafy dyskowe na fragmenty. Pierwszy z nich wyznacza podział na duże fragmenty o małej średnicy. Podział taki minimalizuje liczbę skupień jednocześnie zapewniając możliwość szybkiej komunikacji wewnątrz nich. Drugi z prezentowanych algorytmów dzieli jednostkowy graf dyskowy na fragmenty, których wielkość mieści się w zadanym przedziale.

Rozdział 4 rozważa zagadnienie kolorowania i związany z nim problem przydziału częstotliwości w sieciach radiowych. Celem przydziału częstotliwości radiowych jest wyeliminowanie zjawiska interferencji, czyli nakładania się i tłumienia sygnałów radiowych. W pierwszej części rozdziału rozpatrywany jest problem kolorowania jednostkowych grafów dyskowych w modelu rozproszonym. Przedstawione w niej jest klika algorytmów rozproszonych o różnych współczynnikach aproksymacji optymalnego kolorowania. Druga część poświęcona jest modelowi sieci w której poszczególnie nadajniki mogą zmniejszać swoją moc. Poprzez taką operację znacząco zmniejsza się gęstość sieci a tym samym redukuje się liczba niezbędnych częstotliwości radiowych.

1.2. Podstawowe definicje i model obliczeń

Do sformułowania i analizy algorytmów działających na zdecentralizowanych sieciach potrzebny jest szereg definicji opisujących zarówno strukturę sieci jak i efektywność działających na niej algorytmów. Poniżej zdefiniowany jest precyzyjnie model grafu reprezentującego zdecentralizowane sieci radiowe oraz model obliczeń, na którym działają prezentowane algorytmy. W dalszej części opisane są parametry istotne dla analizy algorytmów przedstawionych w rozprawie.

Jak już zostało wspomniane, rozprawa ta skupia się na klasie sieci informatycznych i telekomunikacyjnych, w których komunikacja odbywa się drogą radiową. Sieci te modelowane są za pomocą grafu dyskowego, w którym wierzchołki odpowiadają elementom sieci rozmieszczonym na płaskim terenie. Przyjmuje się, że dwa wierzchołki łączą się krawędzią jeśli odpowiadające im elementy sieci są w zasięgu swoich nadajników.

Definicja 1.1. *Grafem G nazywamy uporządkowaną parę $G = (V(G), E(G))$ składającą się z niepustego zbioru wierzchołków $V(G)$ oraz zbioru krawędzi $E(G)$ będących nieuporządkowanymi parami wierzchołków z $V(G)$.*

Definicja 1.2. *Jednostkowym grafem dyskowym (UDG ang. Unit Disk Graph) nazywamy graf w którym wierzchołki odpowiadają punktom na płaszczyźnie oraz para wierzchołków v i w tworzy krawędź wtedy i tylko wtedy, gdy $\|v, w\| \leqslant 1$, gdzie przez $\|\cdot, \cdot\|$ oznaczamy odległość w normie euklidesowej.*

W niniejszej pracy przyjęte będzie założenie, że zbiór wierzchołków UDG jest mocy n.

Definicję rozproszonego modelu obliczeń przyjmiemy w formie zaprezentowanej w pracy [59]. Jest ona dostosowana do charakterystyki zdecentralizowanych sieci, gdyż formalizuje obserwację, że pojedynczy element może się komunikować tylko ze swoimi sąsiadami.

Definicja 1.3. *Modelem LOCAL nazywamy rozproszony model obliczeń w którym zakładamy, że:*

1. *Lokalne zegary jednostek obliczeniowych (wierzchołków grafu) są zsynchronizowane, czyli możemy wykonywać obliczenia w rundach.*
2. *W każdej rundzie synchronicznej wierzchołek może wysyłać lub odbierać informacje od sąsiadów i wykonywać lokalne obliczenia.*
3. *Pojedynczy wierzchołek ma informacje tylko na temat sąsiadów (nie zna struktury całego grafu).*
4. *Nie ma ograniczeń na wielkość przesyłanych komunikatów w jednej rundzie.*
5. *Nie ma ograniczonej mocy obliczeniowej pojedynczego wierzchołka.*

Opisane powyżej rundy synchroniczne można interpretować jako kolejne sekundy.

W przypadku algorytmów działających w modelu rozproszonym na grafach dyskowych do powyższej definicji można dodać jeszcze jeden warunek, że każdy wierzchołek zna swoje współrzędne na płaszczyźnie (np. poprzez wbudowany system GPS). Model LOCAL rozszerzony o to założenie nazywać będziemy LOCAL+COORD.

Dodatkowo będziemy zakładać, że każdy wierzchołek posiada unikalny identyfikator ID czyli liczbę z zakresu $0 - poly|V(G)|$, która jest różna dla każdego wierzchołka (przez $poly|V(G)|$ oznaczamy tutaj dowolną funkcję wielomianową zależną od $|V(G)|$). W praktyce identyfikator ID jest to adres MAC, który nadawany jest każdemu urządzeniu. Takie założenie znacząco upraszczać będzie zapis wielu algorytmów, jak np. algorytmu wyboru lidera, czyli wyboru dokładnie jednego wierzchołka w danym grafie.

Ze względu na prostotę definicji, a co za tym idzie i łatwość analizy, model LOCAL jest najczęściej stosowanym modelem obliczeń rozproszonych. Mimo bardzo ogólnych założeń jest to model realistyczny. Po pierwsze, w praktyce lokalne zegary mogą być zsynchronizowane, na przykład przez zewnętrzny zegar sterowany przez system GPS. W dodatku istnieją metody, dzięki którym można przenieść algorytmy stworzone w modelu LOCAL na inne modele, na przykład na model asynchroniczny, w którym zegary nie są zsynchronizowane albo model, w którym wierzchołki mają ograniczone moce obliczeniowe.

Algorytmy działające w modelu LOCAL mogą być analizowane pod względem wielu parametrów. W rozprawie skupimy się na najistotniejszych z nich: czasie działania, długości wysyłanych komunikatów i niezbędnej mocy obliczeniowej wierzchołków. Przez czas działania rozumieć będziemy liczbę synchronicznych rund potrzebnych do rozwiązania problemu. Większość prezentowanych w pracy algorytmów będzie rozwiązywała problem w stałej liczbie rund. Drugim rozpatrywanym parametrem jest długość komunikatów, zdefiniowana jako maksymalna liczba bitów wysyłana w jednej rundzie synchronicznej. Będziemy mówić, że algorytm rozproszony używa *krótkich komunikatów* (ang. short messages), jeśli liczba ta wynosi $O(\log(|V(G)|))$. Podobne ograniczenie nakładać będziemy na moc obliczeniową wierzchołków. Powiemy, że *zakładana moc obliczeniowa jest* $O(f)$ jeśli w jednej rundzie każdy wierzchołek musi wykonać co najwyżej $O(f)$ elementarnych obliczeń.

W związku z tym, że rozwiązanie dokładne problemów rozpatrywanych w pracy może być niemożliwe, interesującą nas klasą algorytmów będą algorytmy przybliżające optymalnie rozwiązanie – algorytmy aproksymacyjne. Do każdego z rozpatrywanych przez nas problemów przyporządkujemy funkcję kosztu (funkcją tą może być np. wielkość wynikowego zbioru, liczba użytych kolorów, sumaryczna waga krawędzi). Celem algorytmów będzie skonstruowanie obiektu, dla którego funkcja kosztu jest możliwie najbliższa funkcji kosztu rozwiązania dokładnego. Po-

wiemy, że algorytm jest k–aproksymacyjny, jeśli funkcja kosztu struktury uzyskanej w wyniku działania algorytmu co najwyżej k krotnie przewyższa optymalną funkcję kosztu. Dla przykładu algorytm 1-aproksymacyjny to algorytm rozwiązujący problem w sposób dokładny.

2. Algorytmy przesyłu informacji

2.1. Wprowadzenie

Jedną z podstawowych funkcji sieci zdecentralizowanych, na przykład sieci sensorowych, jest przesyłanie informacji, czyli rozpowszechnienie pewnej wiadomości wewnątrz sieci. Zagadnienie to zawiera w sobie całe spektrum problemów. W rozprawie ograniczymy się do tych związanych z komunikacją i transmisją danych. Problemy przesyłu informacji można podzielić w zależności od tego ile elementów sieci stanowi źródło wiadomości oraz ile z nich jest adresatem. W szczególności wiadomości mogą być wysyłane i adresowane do jednego, kilku lub wszystkich elementów sieci. Oprócz liczebności zbiorów źródeł i adresatów, o sposobie rozwiązania problemu decydują także założenia o efektywności przesyłu informacji bezpośrednio między elementami sieci. W najprostszym przypadku można przyjąć, że każda informacja wysłana z wierzchołka dociera zawsze do wszystkich jego sąsiadów. Jednakże, ze względu na zastosowania, czasem trzeba założyć, że gdy wierzchołek otrzymuje jednocześnie informacje od dwóch różnych wierzchołków, następuje interferencja uniemożliwiająca mu odbiór tych wiadomości.

Pierwsze zagadnienie, które analizowane jest w tym rozdziale dotyczy przesyłu informacji z jednego wierzchołka do całej sieci w przypadku, gdy nigdy nie zachodzą interferencje. W rozdziale 2.2 przedstawiony jest problem związany ze znalezieniem przy powyższych założeniach struktury minimalizującej sumaryczny koszt rozesłania informacji. Zdefiniowany jest koszt przesyłu informacji przez krawędź i podany algorytm znajdujący obiekt będący bliską aproksymacją rozwiązania tego problemu (tzn. dobrze przybliżający minimalne drzewo rozpinające).

W przypadku, gdy istnieje możliwość powstania interferencji, nie tylko minimalizacja kosztu przesyłu, ale także jego wykonalność i efektywność stają się trudnymi problemami. Dlatego tematem rozdziału 2.3 jest zaprezentowanie algorytmu rozpowszechniania informacji w grafie, gdy zachodzą interferencje. Przedstawiony algorytm konstruuje strukturę propagacji wiadomości z jednego wierzchołka do wszystkich pozostałych. Zaprezentowana jest także jego modyfikacja umożliwiająca implementację w sieci dynamicznej z pojawiającymi się i znikającymi wierzchołkami. Poza tym opisane są wersje algorytmu umożliwiające przesył informacji

z wszystkich do wszystkich wierzchołków jednocześnie oraz wysłanie w pewnych odstępach czasu kilku informacji z jednego źródła. Wszystkie powyżej opisane wyniki dotyczą przypadku, gdy wszystkie elementy sieci wykorzystują do nadawania informacji jedną częstotliwość. W ostatniej części rozdziału rozważany jest problem przesyłu informacji w przypadku, gdy wierzchołki mogą nadawać na różnych częstotliwościach.

Wyniki zawarte w tym rozdziale częściowo oparte są na pracach [41] i [44].

2.2. Minimalne drzewo rozpinające

W sieciach radiowych każde połączenie i transmisja danych wiąże się z pewnym kosztem energetyczny. Koszt ten zależy znacząco od odległości elementów, które się ze sobą komunikują. Przyjmując jako model sieci model grafowy, koszt komunikacji opisujemy w terminach wagi krawędzi. W sieciach radiowych jako wagę krawędzi przeważnie przyjmuje się odległość bądź kwadrat odległości między elementami. Propagacja informacji od wybranego wierzchołka do całej sieci wiąże się z dużą ilością transmisji pomiędzy wieloma wierzchołkami. Naturalnym jest więc pytanie o taką strukturę propagacji informacji, w której sumaryczny koszt transmisji będzie zminimalizowany.

W terminach teorii grafów taka struktura nazywa się minimalnym drzewem rozpinającym. Poprzez minimalne drzewo rozpinające grafu G rozumiemy spójny podgraf grafu G o zbiorze wierzchołków $V(G)$, minimalnej wadze i nie zawierający cykli (czyli minimalnej sumie wag krawędzi w nim zawartych). Tak więc minimalne drzewo rozpinające (MST ang. Minimal Spanning Tree) określa, które połączenia (krawędzie) użyć podczas propagacji wiadomości, aby wszystkie wierzchołki odebrały informację, a jednocześnie sumaryczny koszt energetyczny połączeń był zminimalizowany.

Problem znalezienia MST w sposób rozproszony w dowolnym grafie (tzn. niekoniecznie w UDG) rozpatrywany był między innymi w pracach [9, 22, 26, 28, 60], jak również w [2, 5, 54, 48].

W niniejszym rozdziale przedstawiony jest rozproszony algorytm AlmostMST, który konstruuje graf będący $(1+O(1/d))$–aproksymacją minimalnego drzewa rozpinającego w jednostkowych grafach dyskowych w czasie $O(d^2)$ rund synchronicznych, gdzie d jest wejściowym parametrem. Algorytm ten wyznacza podgraf K jednostkowego grafu dyskowego G taki, że K zawiera minimalne drzewo rozpinające grafu G. Co więcej K jest planarny, nie zawiera cykli o długości mniejszej niż $d/3$, a jego waga stanowi $(1+O(1/d))$-aproksymację wagi minimalnego drzewa rozpinającego G.

Punktem wyjścia w konstrukcji jest rozproszony algorytm ApproxMST przedstawiony w [48]. Znajduje on planarny podgraf L jednostkowego grafu dyskowego G. Graf L zawiera MST grafu G jako podgraf oraz ma maksymalny stopień ograniczony przez stałą. Sumaryczna waga wszystkich krawędzi w L jest równa wadze minimalnego drzewa rozpinającego G pomnożonej przez pewną stałą (wynoszącą 40). Konstrukcja grafu L bazuje na koncepcji grafu relatywnego sąsiedztwa (ang. relative neighborhood graph (RNG)) i zajmuje stałą liczbę kroków synchronicznych.

ApproxMST (z pracy [48])
Wejście: Jednostkowy graf dyskowy G
Wyjście: Podgraf L grafu G. Podgraf L jest planarny, ma ograniczony stopień a jego waga jest stałą aproksymacją wagi $MST(G)$.

Prezentowany algorytm AlmostMST korzysta z grafu L wygenerowanego przez algorytm ApproxMST, jako wejścia. Bazując na jego własnościach, znajduje w nim podgraf K, którego waga jest $1 + O(1/d)$ aproksymacją wagi MST. W naszym podejściu kładziemy nacisk na poprawę współczynnika aproksymacji algorytmu AlmostMST kosztem pogorszenia innych parametrów. I tak, nasz algorytm działa w czasie $poly(d)$ rund synchronicznych (d jest z góry daną stałą) podczas, gdy ApproxMST w czasie $O(1)$. Również parametry lokalności i kosztu komunikacji zwiększają się w porównaniu z algorytmem ApproxMST. Dokładniej mówiąc, zamiast informacji pobieranej tylko z dwu-sąsiedztwa, Algorytm AlmostMST korzysta z informacji z z $O(d^2)$-sąsiedztwa. W dodatku koszt komunikacji w przedstawionym algorytmie, dla grafu na n wierzchołkach, wynosi $O(n^2)$ (dla porównania $O(n)$ w algorytmie ApproxMST).

W dalszej części rozdziału najpierw zaprezentujemy główny algorytm, podstawowe definicje oraz ideę metody trzech siatek, na której opiera się algorytm. Następnie pokażemy, że graf K zadany przez AlmostMST zawiera $MST(G)$ jako podgraf, nie zawiera krótkich cykli a jego waga jest dobrą aproksymacją wagi MST(G). Rozdział zakończymy dowodem, że algorytm działa w czasie $poly(d)$ synchronicznych rund.

Przypomnijmy, że prezentowany algorytm pracuje w rozproszonym modelu obliczeń LOCAL+COORD (patrz definicja 1.3 z uwagami) na jednostkowym grafie dyskowym G o zbiorze wierzchołków V ($|V| = n$) i zbiorze krawędzi E (patrz definicja 1.2). Przez wagę krawędzi ($e = (u, v)$, $u, v \in V$) w jednostkowym grafie dyskowym rozumieć będziemy odległość pomiędzy v a u i oznaczać będziemy przez $w(e) = \|v, u\|$. Jeśli $H \subseteq G$ jest podgrafem grafu G, wtedy wagę grafu H zdefiniujmy jako $w(H) = \sum_{e \in E(H)} w(e)$. Dla danego spójnego grafu G, przez MST(G)

oznaczmy drzewo T rozpinające graf G takie, że waga $w(T)$ jest najmniejsza spośród wszystkich drzew rozpinających G.

Główną ideą przedstawionego algorytmu jest użycie metody trzech przecinających się siatek. W technice tej dzielimy płaszczyznę trzema różnymi siatkami, które z kolei definiują trzy klasy kwadratów. W pierwszym kroku obliczamy lokalnie optymalne rozwiązania w kwadratach pierwszej siatki. W następnym kroku poprawiamy te rozwiązania wewnątrz kwadratów drugiej siatki a następnie wykonujemy ten sam zabieg w kwadratach wewnątrz siatki trzeciej. Analogiczna idea przemieszczania siatek została niezależnie zaproponowana przez Lillis, Pemmaraju i Pirwani w pracy [50].

Rozpatrzmy siatkę $L_d^{(0,0)}$ przechodzącą przez punkt $(0,0)$ i zawierającą poziome i pionowe linie odległe od siebie o d oraz dwie inne siatki $L_d^{(d/3,d/3)}$ i $L_d^{(2d/3,2d/3)}$ otrzymane z $L_d^{(0,0)}$ poprzez przesunięcie jej o wektory $(d/3, d/3)$ i $(2d/3, 2d/3)$, odpowiednio (patrz rysunek 2.1). Zbiór kwadratów ograniczonych prostymi należącymi do siatki $L_d^{(\nu_1,\nu_2)}$ oznaczmy przez $\mathcal{S}_d^{(\nu_1,\nu_2)}$. Oznaczmy też przez $G[\mathcal{S}]$ podgraf grafu G indukowany na wierzchołkach zawartych w kwadracie $\mathcal{S}$ (czyli wierzchołki ze zbioru $\{v \in V(G) : v \in \mathcal{S}\}$).

Niech H będzie podgrafem grafu G złożonym z k spójnych składowych oznaczonych przez $H_1, H_2, \ldots H_k$. Zdefiniujmy procedurę SpanSub następująco:

SpanSub(G, H)
Wejście: Graf G i jego podgraf H.
Wyjście: Podgraf S rozpinający graf G
1. Podstaw $V(S) := V(G)$
2. Podstaw $E(S) := (E(G) \setminus E(H)) \cup \left(\bigcup_{i=1}^{k} E(\mathrm{MST}(H_i))\right)$
3. Zwróć S jako wynik.

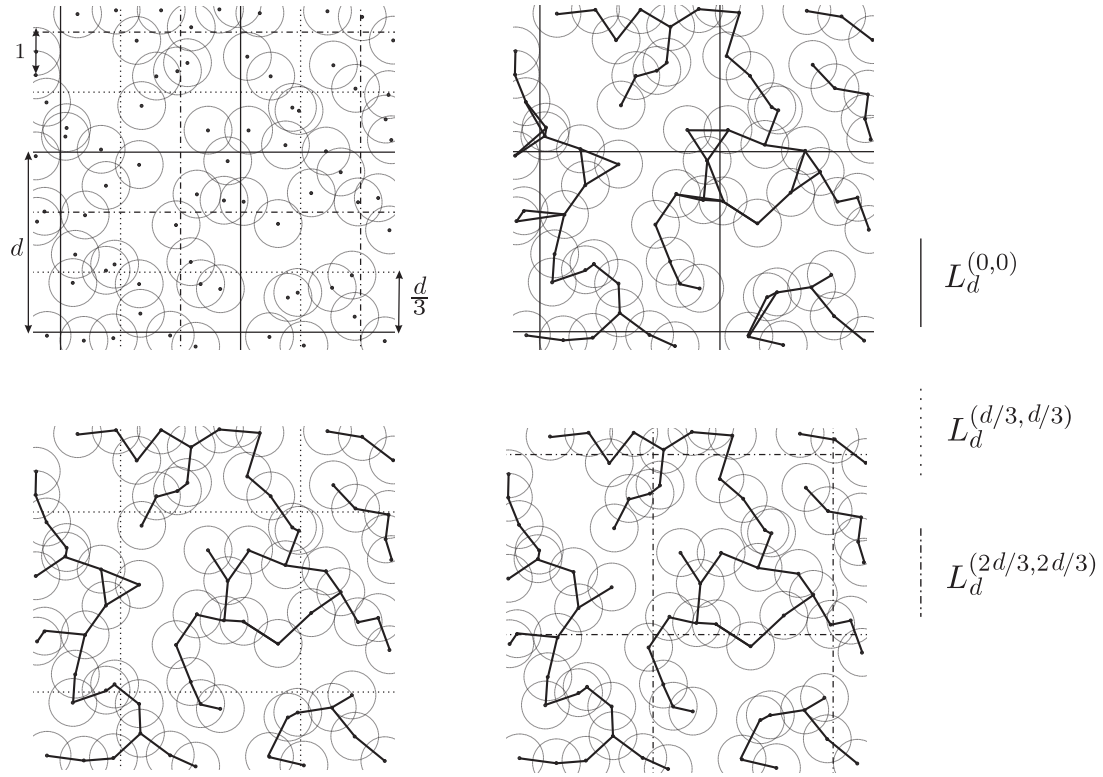
Przedstawiony poniżej główny algorytm AlmostMST wielokrotnie będzie wywoływać lokalnie procedurę SpanSub(G, H).

W poniższym twierdzeniu podsumujemy wszystkie własności grafu wynikowego, wygenerowanego przez algorytm AlmostMST.

Twierdzenie 2.1. *Niech G będzie jednostkowym grafem dyskowym i niech $d > 9$ będzie wybranym parametrem. Niech K będzie grafem generowanym przez algorytmu* AlmostMST. *Wtedy K jest planarnym grafem o ograniczonym stopniu, $MST(G) \subseteq K$, $w(K) \leqslant (1 + \frac{6}{d-6})w(MST(G))$. Ponadto waga każdego cyklu w K wynosi co najmniej $d/3$.*

Udowodnimy twierdzenie 2.1 poprzez ciąg lematów.

AlmostMST
Wejście: Spójny jednostkowy graf dyskowy G i z góry dany parametr d .
Wyjście: Graf K zawierający pewne MST(G) .
(1) Uruchom algorytm ApproxMST na grafie G. Wyjściowy graf oznacz jako L.
(2) Uruchom równolegle na każdym kwadracie SpanSub$\Big(L, \bigcup_{\mathcal{S}\in\mathcal{S}_d^{(0,0)}} L[\mathcal{S}]\Big)$. Oznacz wynikowy graf przez N'.
(3) Uruchom równolegle SpanSub$\Big(N', \bigcup_{\mathcal{S}\in\mathcal{S}_d^{(d/3,d/3)}} N'[\mathcal{S}]\Big)$. Oznacz wynikowy graf jako N''.
(4) Uruchom równolegle SpanSub$\Big(N'', \bigcup_{\mathcal{S}\in\mathcal{S}_d^{(2d/3,2d/3)}} N''[\mathcal{S}]\Big)$. Zwróć wynikowy graf K.



Rysunek 2.1. Idea działania algorytmu AlmostMST.

Lemat 2.2. *Niech G będzie spójnym jednostkowym grafem dyskowym. Wynikowy graf K algorytmu* AlmostMST *jest planarny, ma ograniczony stopień i zawiera MST(G).*

Dowód. Planarność i ograniczony stopień grafu K są konsekwencją faktu, że K jest podgrafem grafu L. Przypomnijmy, że graf L wygenerowany jest przez algorytm ApproxMST i posiada własności planarności i ograniczonego stopnia (patrz praca [48]). Co więcej L zawiera $\mathrm{MST}(G)$. Zostaje więc jedynie do pokazania, że K również zawiera $\mathrm{MST}(G)$ jako podgraf. Bez straty ogólności załóżmy, że wszystkie krawędzie mają różne wagi co oznacza, że $\mathrm{MST}(G)$ jest wyznaczony jednoznacznie.

Zauważmy, że jeśli H jest spójnym podgrafem spójnego grafu G, to zachodzi $\mathrm{MST}(G) \subseteq \textsc{SpanSub}(G, H)$ co jest równoważne ze stwierdzeniem, że zachodzi $E(\mathrm{MST}(G)) \subseteq (E(G) \setminus E(H)) \cup (E(MST(H)))$.

Załóżmy że tak nie jest. Oznacza to, że istnieje krawędź $e = uv$ taka, że $e \in (E(\mathrm{MST}(G)) \cap E(H)) \setminus E(\mathrm{MST}(H))$. Ponieważ H jest spójny, to istnieje ścieżka w $\mathrm{MST}(H)$ łącząca wierzchołki u i v. Oznaczmy ją przez $p_{u,v}$. Krawędź $e \notin \mathrm{MST}(H)$ stąd $\forall_{e' \in p_{u,v}} w(e') < w(e)$. W przeciwnym razie możemy zamienić pewną krawędź z $p_{u,v}$ na krawędź e i utworzyć drzewo rozpinające o wadze mniejszej niż $MST(G)$. Rozpatrzmy więc graf $M = \mathrm{MST}(G) \setminus e$, który zawiera dwie składowe spójności M_u , zawierającą wierzchołek u, i M_v, zawierającą wierzchołek v. Co najmniej jedna krawędź $e' \in p_{u,v}$ łączy M_u z M_v. Stąd możemy skonstruować drzewo rozpinające $M' = (\mathrm{MST}(G) \cup e') \setminus e$ takie, że $w(M') < \mathrm{MST}(G)$ (założyliśmy, że $w(e') < w(e)$), a to stanowi sprzeczność. □

Poniższa obserwacja będzie pomocna w dowodzie lematu 2.4.

Fakt 2.3. *Dla dowolnych trzech punktów $q_0 \in L_a^{(0,0)}$, $q_1 \in L_a^{(a/3,a/3)}$ i $q_2 \in L_a^{(2a/3,2a/3)}$ mamy $\|q_0, q_1\| + \|q_0, q_2\| \geqslant a/3$.*

Dowód. Niech $\left\|q_0, L_a^{(\nu_1,\nu_2)}\right\| = \min_{q' \in L_a^{(\nu_1,\nu_2)}} \|q_0, q'\|$. Udowodnimy lemat dla dowolnego punkt $q_0 = (x, y)$ leżącego na odcinku o końcach $(0, 0)$ i $(0, a)$. W tym przypadku mamy:

$$\left\|q_0, L_a^{(a/3,a/3)}\right\| = \begin{cases} a/3 - y & \text{jeśli } y \in [0, a/3] \\ y - a/3 & \text{jeśli } y \in [a/3, 2a/3] \\ a/3 & \text{jeśli } y \in [2a/3, a] \end{cases}$$

$$\left\|q_0, L_a^{(2a/3,2a/3)}\right\| = \begin{cases} a/3 & \text{jeśli } y \in [0, a/3] \\ 2a/3 - y & \text{jeśli } y \in [a/3, 2a/3] \\ y - 2a/3 & \text{jeśli } y \in [2a/3, a]\,. \end{cases}$$

Stąd uzyskujemy

$$\left\|q_0, L_a^{a/3,a/3)}\right\| + \left\|q_0, L_a^{(2a/3,2a/3)}\right\| = \begin{cases} 2a/3 - y & \text{jeśli } y \in [0, a/3] \\ a/3 & \text{jeśli } y \in [a/3, 2a/3] \\ y - a/3 & \text{jeśli } y \in [2a/3, a]\,. \end{cases}$$

Tak więc w rozważanym przedziale suma odległości jest co najmniej $a/3$. Dla innych przedziałów dowód przebiega analogicznie. □

Lemat 2.4. *Wszystkie cykle w wynikowym grafie K algorytmu* AlmostMST *mają wagę co najmniej $d/3$.*

Dowód. Fakt ten bezpośrednio wynika z obserwacji, że jeśli C jest cyklem w grafie K, wtedy (traktowany jako krzywa zamknięta) musi przecinać wszystkie trzy siatki $L_d^{(0,0)}$, $L_d^{d/3,d/3)}$ oraz $L_d^{(2d/3,2d/3)}$. By tego dowieść załóżmy, że C nie przecina $L_d^{(0,0)}$, czyli jest całkowicie zawarty w $\mathcal{S} \in \mathcal{S}_d^{(0,0)}$. To jest jednak niemożliwe, ponieważ co najmniej jedna z krawędzi zostałaby usunięta z C podczas pierwszego kroku algorytmu AlmostMST. Analogiczny argument dowodzi, że C musi przecinać dwie pozostałe siatki. Tak więc niech q_0, q_1 i q_2 będą punktami, w których C (traktowany jako krzywa) przecina $L_d^{(0,0)}$, $L_d^{(d/3,d/3)}$ i $L_d^{(2d/3,2d/3)}$. Korzystając z faktu 2.3, uzyskujemy $w(C) \geqslant \|q_0, q_1\| + \|q_0, q_2\| \geqslant d/3$. □

Lemat 2.5. *Wynikowy graf K algorytmu* AlmostMST *spełnia warunek:*

$$w(K) \leqslant (1 + 6/(d-6))w(MST(G)).$$

Dowód. Ponieważ K jest podgrafem grafu L wygenerowanego przez algorytm ApproxMST, stąd K jest planarny. Oznaczmy przez $|F(K)|$ liczbę ograniczonych ścian w płaskim zanurzeniu K na płaszczyźnie. Każda ograniczona ściana odpowiada pewnemu cyklowi C. Korzystając z lematu 2.4, każdy cykl C ma wagę co najmniej $d/3$. Ponieważ każda krawędź w C należy do wnętrza co najwyżej dwóch ścian, stąd:

$$w(K) \geqslant \frac{|F(K)|}{2}\frac{d}{3}.$$

Z lematu 2.2 wynika, że K zawiera $MST(G)$. Ponadto ze wzoru Eulera mamy $|V(K)| - |E(K)| + |F(K)| = 1$. $|F(K)|$ oznacza liczbę ograniczonych ścian w płaskim zanurzeniu grafu G. Tak więc $MST(G)$ może być otrzymane z K poprzez usunięcie dokładnie $|F(K)|$ krawędzi. Ponieważ każda usuwana krawędź ma wagę co najwyżej 1 zatem:

$$\frac{|F(K)|\,d}{6} \leqslant w(K) \leqslant |F(K)| + w(\mathrm{MST}(G)).$$

Stąd,

$$|F(K)| \leqslant \frac{6w(\mathrm{MST}(G))}{d-6},$$

co z kolei implikuje, że

$$w(K) \leqslant \left(1 + \frac{6}{d-6}\right) w(\mathrm{MST}(G)).$$

□

BreadthFirstSearch(G, H)
Wejście: Graf G i wyróżniony wierzchołek (lider) s posiadający wiadomość m_s.
Wyjście: Graf G w którym wszystkie wierzchołki posiadają wiadomość m_s.
1. W pierwszej rundzie synchronicznej s otrzymuje indeks 1 reszta wierzchołków otrzymuje indeks 0.
2. W i-tej rundzie synchronicznej ($i > 1$) wszystkie wierzchołki o indeksie $i - 1$ przekazują wiadomość m_s sąsiadom o indeksie 0.
3. Wszystkie wierzchołki które w i-tej rundzie otrzymały wiadomość m_s otrzymują indeks i.

Na koniec pokażemy, że algorytm AlmostMST w modelu LOCAL+COORD może być zrealizowany w czasie $poly(d)$ rund synchronicznych. Implementacja ta bazuje na algorytmie FindingMstInSquare opisanym poniżej i ma taką samą jak on złożoność czasową.

Twierdzenie 2.6. *Algorytm* AlmostMST *działa w $O(d^2)$ synchronicznych rundach w modelu obliczeń LOCAL+COORD.*

W algorytmie FindingMstInSquare będziemy wykorzystywać procedurę przeszukiwania wszerz. Rozproszona procedura przeszukiwania wszerz służy do zebrania informacji o całym grafie w jednym wierzchołku lub też przekazania przez jeden wierzchołek informacji całemu grafowi. Poniżej przestawimy jedną z wielu wersji tej procedury:

Nie trudno zauważyć, że wierzchołek o indeksie i znajduje się w odległości $i - 1$ od s. Czas działania tego algorytmu wynosi $O(diam(G))$, gdzie $diam(G)$ oznacza średnicę grafu czyli największą możliwą odległość (długość najkrótszej ścieżki) pomiędzy parą wierzchołków w G.

Przedstawmy teraz algorytm FindingMstInSquare który jest implementacją procedury SpanSub działającej na wierzchołkach UDG wewnątrz kwadratu.

Lemat 2.7. *Niech d będzie długością boku kwadratu $\mathcal{S}$. Algorytm* FindingMstInSquare *działa w czasie $O(d^2)$ synchronicznych rund w modelu obliczeń LOCAL+COORD.*

Dowód. Najpierw zauważmy, że kroki (1), (2) i (4) algorytmu FindingMstInSquare zajmują $O(diam(H[\mathcal{S}]))$ synchronicznych rund, gdzie $diam(H[\mathcal{S}])$ oznacza średnicę grafu $H[\mathcal{S}]$. Ponieważ w modelu LOCAL nie nakładamy ograniczeń na moc obliczeniową wierzchołków więc krok (3) algorytmu FindingMstInSquare zajmuje jedną synchroniczną rundę. Zauważmy, że graf H' jest podgrafem grafu

FindingMstInSquare
Wejście: Spójna składowa H' grafu $H[\mathcal{S}]$, gdzie H jest podgrafem jednostkowego grafu dyskowego G a $\mathcal{S}$ jest kwadratem o boku d.
Wyjście: MST(H') .
(1) Zostaje wyznaczony lider (czyli wyróżniony wierzchołek) w H'.
(2) Uruchomiona zostaje procedura BreadthFirstSearch na grafie $G[\mathcal{S}]$. Przy jej użyciu dostarczana jest liderowi całą informację na temat grafu H' i wag krawędzi w nim zawartych.
(3) Lider grafu H', używając klasycznego algorytmu Kruskala, wylicza lokalnie MST(H').
(4) Lider H', ponownie używając procedury BreadthFirstSearch, wysyła wierzchołkom z H' informacje na temat tego, które krawędzie zawarte są w MST(H').

L, a zatem ma stopień ograniczony przez stałą. Stąd problem lokalnego wyznaczenia MST(H') ma złożoność obliczeniową rzędu $|V(H')|^2$ (czyli zakładana moc obliczeniowa w jeden rundzie musi być rzędu $|V(H')|^2$).

Pozostaje wykazać, że jeśli $H' \subseteq H[\mathcal{S}]$ jest spójny, to $diam(H') = O(d^2)$. W tym celu oznaczmy przez $c(r, v)$ koło o promieniu r i środku w wierzchołku v. Weźmy dwa różne wierzchołki $w_i, w_j \in \text{MIS}(H')$, gdzie MIS oznacza największy zbiór niezależny grafu H'. Wtedy zachodzi nierówność $\|w_i, w_j\| > 1$, zatem koła $c\left(\frac{1}{2}, w_i\right)$ i $c\left(\frac{1}{2}, w_j\right)$ są rozłączne. Zauważmy, że każde $c\left(\frac{1}{2}, w_i\right)$ leży w kwadracie o boku $d+1$, a taki kwadrat zawiera co najwyżej $\frac{(d+1)^2}{\pi/4}$ rozłącznych kół o promieniu $\frac{1}{2}$. Oznacza to że wielkość MIS(H') wynosi co najwyżej $\frac{4(d+1)^2}{\pi}$. Niech $v, w \in G[\mathcal{S}]$ będą dwoma wierzchołkami takimi, że najkrótsza ścieżka $p_{v,w}$ pomiędzy v a w ma długość równą średnicy H'. Zauważmy, że co drugi wierzchołek na takiej ścieżce tworzy zbiór niezależny. Stąd $diam(H') \leqslant 2|\text{MIS}(H')| \leqslant \frac{8(d+1)^2}{\pi}$. □

2.3. Rozpowszechnianie informacji bez kolizji

W tym rozdziale rozpatrzymy jeden z podstawowych problemów sieci ad hoc – bezkolizyjne rozpowszechnianie informacji gdy wybrany wierzchołek wysyła wiadomość do wszystkich wierzchołków. Jeśli podczas rozpowszechniania informacji dwa wierzchołki przesyłają wiadomość jednocześnie do tego samego wierzchołka, wówczas zachodzi interferencja i wiadomość nie zostaje odebrana. Głównym celem bezkolizyjnego rozpowszechniania informacji jest zminimalizowanie liczby rund potrzebnych do rozesłania informacji z uwzględnieniem kolizji. Problem ten znany jest pod nazwą MLBS (ang. minimum latency broadcast schedule).

W niniejszym rozdziale zaprezentujemy rozproszony algorytm znajdujący stałą aproksymację problemu MLBS w grafach dyskowych. Przedstawiona konstrukcja zajmuje stałą liczbę rund synchronicznych i używa tylko krótkich komunikatów. Co istotne, działanie prezentowanego algorytm jest niezależne od wyboru wierzchołka rozsyłającego wiadomość i nie wymaga zbudowania drzewa BFS. Dzięki temu uzyskuje się możliwość szybkiej adaptacji i zmiany struktury rozpowszechniania informacji. Oznacza to, że podczas przesyłu wiadomości następować mogą zdarzenia takie jak dodawanie, przesuwanie bądź usuwanie wierzchołków.

W rozdziale tym rozpatrzymy również problem plotkowania (ang. minimum – latency gossiping), czyli zagadnienia, w którym każdy wierzchołek ma unikalną informację, którą przekazuje pozostałym wierzchołkom. Przedstawimy algorytm rozproszone w modelu z nieograniczoną wielkością komunikatów rozwiązujący problem plotkowania w UDG.

Przedstawimy również zastosowania zaprezentowanej konstrukcji do rozwiązania problemu rozpowszechnienia wielu wiadomości (ang. single source multiple message broadcasting) oraz problemu wielokanałowego rozpowszechnienia wiadomości (multi channel broadcast scheduling)

Zaprezentowany algorytm rozproszony działa przy założeniu, że komunikacja w bezprzewodowych sieciach jest zsynchronizowana i składa się z ciągu rund przesyłu informacji w modelu LOCAL+COORD. Zakładamy ponadto, że wiadomość nadawana przez dany wierzchołek jest zawsze wysyłana do wszystkich sąsiadów. Oznacza to, że nie można wybrać tylko podzbioru sąsiadów do których dany wierzchołek nadaje. W każdej rundzie synchronicznej wierzchołek v albo nadaje wiadomość albo stara się wiadomość odebrać od sąsiada. Jednakże fakt, że jakiś wierzchołek z sąsiedztwa v wysłał wiadomość wcale nie musi oznaczać iż v ją odebrał. Jeśli kilku sąsiadów v nadawało jednocześnie mogło nastąpić zniekształcenie sygnału bądź jego wzajemne wytłumienie. Dlatego mówimy, że w otrzymał wiadomości od v w rundzie i-tej, jeśli w trakcie tej rundy wierzchołek v słuchał a wierzchołek w był jedynym z sąsiadów v który nadawał wiadomość.

Jak już wspomnieliśmy pierwszym problemem który rozważymy jest jak najszybsze rozpowszechnienie informacji wysłanej przez dany wierzchołek s. Wierzchołek s nazywany będziemy źródłem (ang. source). Rozwiązanie problemu polega na utworzeniu ciągu kolejnych przekazujących informację grup wierzchołków, tak by po jak najmniejszej liczbie rund każdy wierzchołek otrzymał wiadomość nadaną przez s. Liczbę tych rund nazywamy latencją i oznaczamy jako l.

Definicja 2.8. *Strukturą rozpowszechniania informacji o latencji l (ang. broadcast schedule of latency l) nazywamy ciąg podzbiorów $(U_1, U_2 \dots U_l)$ zbioru $V(G)$ spełniający warunki:*

(1) $U_1 = \{s\}$*;*
(2) $U_i \subseteq \bigcup_{j=1}^{i-1} Inf(U_j)$ *dla każdego* $2 \leqslant i \leqslant l$*;*
(3) $V(G) \setminus \{s\} \subseteq \bigcup_{j=1}^{l} Inf(U_j)$*,*

gdzie $Inf(U)$ *oznacza zbiór takich wierzchołków spośród* $V(G) \setminus U$ *z których każdy posiada dokładnie jednego sąsiada w* U*.*

Inaczej rzecz ujmując, $Inf(U)$ to zbiór tych wierzchołków które otrzymały wiadomość nadaną przez wierzchołki z U. Tak więc, strukturą rozpowszechniającą informację jest ciąg podzbiorów wierzchołków $(U_1, U_2 \dots U_l)$ taki, że jeśli w i-tej rundzie nadają tylko wierzchołki z U_i to w l-tej rundzie wszystkie wierzchołki będą dysponować informacją nadaną ze źródła s.

W pracy tej zajmiemy się zarówno minimalizowaniem latencji l, jak również minimalizowaniem liczby synchronicznych rund potrzebnych do znalezienia ciągu $(U_1, U_2 \dots U_l)$. Dla dowolnego grafu G (nie koniecznie grafu dyskowego) najlepsze znane algorytmy budują strukturę rozpowszechniania informacji o latencji odpowiednio $R + O(\log^3 n / \log\log n)$ (praca Cicalese, Manne i Xin [21]) oraz $O(R + \log^2 n)$ (praca Kowalski i Pelc [38]) gdzie R oznacza promień grafu, czyli maksymalną odległość pomiędzy s a wierzchołkami z $V(G)$, a n to liczba wierzchołków w grafie ($n = |V(G)|$). Inne wyniki dotyczące rozpowszechniania informacji mają współczynnik latencji odpowiednio równy : $O(R\Delta)$ [10] , $O(R\log^2(n/R))$ [11] , $O(R\log n + \log^2 n)$ [37] , $R + O(\sqrt{R}\log^2 n)$ [53], $O(R + \log^6 n)$ [25] oraz $R + O(\log^3 n)$ [46].

Problem rozpowszechniania informacji w jednostkowych grafach dyskowych rozważany był w pracach [18],[27],[35] i [36]. W pracy [18] Dessmark i Pelc zaprezentowali algorytm rozpowszechniania informacji o latencji $2400R$. Gandhi, Parthasarathy i Mishra w [27] udowodnili NP-trudność problemu MLBS w grafach dyskowych i stworzyli strukturę rozpowszechniającą informację o latencji $648R$. Z kolei w pracy [35] Scott, Huang, Wan, Jia i Du przedstawili trzy algorytmy o latencjach odpowiednio $51R$, $24R$, i $R + O(\sqrt{R}\log^{1.5} R)$. Najlepszy dotychczasowy wynik w tej dziedzinie ustanowili Huang, Wan, Jia, Du i Shang w pracy [36], gdzie zaprezentowano trzy algorytmy o współczynnikach latencji odpowiednio $24R + 23$, $16R + 15$ i $R + O(\log R)$.

Znaczącą słabością tych wyników jest to, że wszystkie przedstawione w nich algorytmy rozpatrywane są w klasycznym (nie rozproszonym) modelu obliczeń. Algorytmy te ściśle bazują na umiejscowieniu źródła informacji i wykorzystują konstrukcję drzewa BFS. Dlatego ewentualna przebudowa struktury rozpowszechniającej informację, związana np. z dynamicznymi zmianami w zdecentralizowanej sieci, jest długotrwała i nieefektywna.

W niniejszym rozdziale zaprezentujemy algorytm HexagonsBroadcasting, który jest pierwszym rozproszonym algorytmem rozwiązującym problem MLBS

bez wykorzystania konstrukcji drzewa BFS. Dokładniej, algorytm HexagonsBroadcasting potrzebuje jedynie $O(1)$ synchronicznych rund, by stworzyć strukturę rozpowszechniającą informację. Wszystkie poprzednie algorytmy potrzebowały na to $O(R)$ rund. Latencja analizowanego algorytmu wynosi $258R$. Struktura wygenerowana przez algorytm HexagonsBroadcasting nie zależy od źródła informacji, przez co źródło może być łatwo zmienione bądź nawet wybrane losowo. Wcześniejsze algorytmy nie brały pod uwagę możliwości zmian w sieci podczas propagacji sygnału. W algorytmach tych, gdy jeden z wierzchołków zostaje usunięty w trakcie rozpowszechniania informacji, może się zdarzyć, że duża frakcja elementów sieci nigdy nie otrzyma wiadomości, mimo iż sieć jest nadal spójna. W naszym algorytmie negatywny wpływ usuwania, dodawania lub przesuwania wierzchołków może być szybko neutralizowany poprzez odświeżanie struktury rozpowszechniania informacji. Metoda ta opisana jest w algorytmie DynamicBroadcasting.

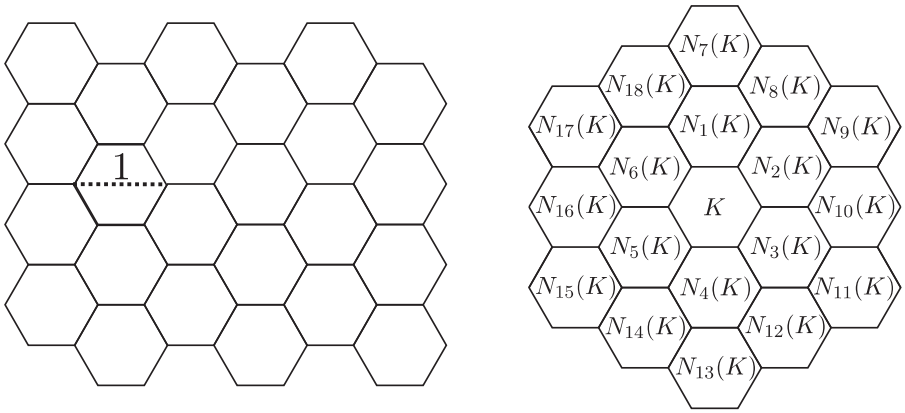
Algorytm HexagonsBroadcasting może być również użyty do rozwiązania kilku innych problemów, takich jak: plotkowanie (gossiping), rozpowszechnienia wielu wiadomości (ang. single source multiple message broadcasting) oraz problemu wielokanałowego rozpowszechnienia wiadomości (multi channel broadcast scheduling).

W rozdziale tym najpierw zaprezentujemy podstawowe definicje i fakty. Następnie przedstawimy dwa pomocnicze algorytmy SelectBroadcastNodes i BroadcastSets oraz główny algorytm HexagonsBroadcasting wraz z dowodem jego poprawności. W osobnym paragrafie zaprezentowany zostanie algorytm GossipingInUDG rozwiązujący problem plotkowania. Na koniec pokażemy dwa uogólnienia problemu MLBS.

Przedstawiony algorytm działa na jednostkowym grafie dyskowym G (patrz definicja 1.2) o zbiorze wierzchołków $V = V(G)$ ($|V| = n$) w rozproszonym, synchronicznym modelu LOCAL+COORD (patrz definicja 1.3).

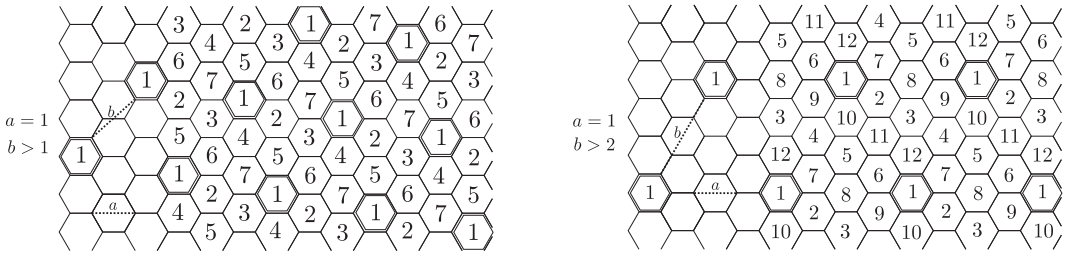
W głównym algorytmie HexagonsBroadcasting używać będziemy podziału zabioru wierzchołków jednostkowego grafu dyskowego bazującego na siatce dzielącej płaszczyznę na heksagony. W siatce tej każdy heksagon ma bok długości 1/2 (patrz rysunek 2.2). Jeśli każdy wierzchołek zna swoje współrzędne na płaszczyźnie, to zna również heksagon do którego należy. Oznaczmy jako G' podgraf grafu dyskowego G taki, że $V(G') = V(G)$ i $vw \in E(G')$ wtedy i tylko wtedy gdy v i w leżą w tym samym heksagonie. Zdefiniujmy $Hex(G)$ jako zbiór wszystkich składowych spójności grafu G'. Zauważmy, że każdy heksagon leży wewnątrz koła o promieniu $\frac{1}{2}$, a zatem każda składowa $K \in Hex(G)$ jest kliką.

Niech $K \in Hex(G)$ będzie pewną kliką. Zauważmy, że wierzchołki z K mogą łączyć się tylko z wierzchołkami należącymi do osiemnastu heksagonów (patrz rysunek 2.2). Oznaczmy te sąsiednie heksagony przez $N_1(K), N_2(K), \ldots, N_{18}(K)$



Rysunek 2.2. Podział płaszczyzny na heksagony oraz numeracja 18-tu sąsiadujących heksagonów.

według rysunku 2.2 (numer kliki zależy od jej umiejscowienia na siatce heksagonalnej). Zauważmy, że zachodzą pewne zależności takie jak np. $N_2(N_5(K)) = K$ i $N_{11}(N_{17}(K)) = K$. Stąd dla każdego $k \in \{1, \ldots, 18\}$ możemy zdefiniować $\bar{k}$ jako liczbę taką, że $N_{\bar{k}}(N_k(K)) = K$ (np. $\bar{5} = 2$, $\bar{17} = 11$). Dodatkowo zdefiniujmy $N(K) = \{K' \in Hex(G) : K' \neq K, \exists_{v \in K, v' \in K'} vv' \in E(G)\}$. Miejmy na uwadze, że $N(v)$ oznacza sąsiadów wierzchołka v, a $N(K)$ kliki sąsiadujące z heksagonem K.



$f^{(7)} : Hex(G) \to \mathbb{N}$ $\qquad$ $f^{(12)} : Hex(G) \to \mathbb{N}$

Rysunek 2.3. Idea funkcji $f^{(7)}$ i $f^{(12)}$.

Zdefiniujmy również dwie funkcje $f^{(7)} : Hex(G) \to \{1, \ldots, 7\}$ oraz $f^{(12)} : Hex(G) \to \{1, \ldots, 12\}$ w sposób przedstawiony na rysunku 2.3. Zauważmy, że z własności funkcji $f^{(7)}$ wynika, że dowolne dwa wierzchołki z różnych heksagonów o tej samej wartości funkcji $f^{(7)}$ są w odległości większej niż 1. Podobnie numeracja heksagonów liczbami $\{1, \ldots, 12\}$ danymi przez funkcję $f^{(12)}$ ma tą własność, że dowolne dwa wierzchołki z różnych heksagonów o tym samym numerze są w odległości większej niż 2. Bazując na tych obserwacjach otrzymujemy:

Fakt 2.9. *Każde dwa wierzchołki z różnych klik $K_1, K_2 \in Hex(G)$ takich, że $f^{(7)}(K_1) = f^{(7)}(K_2)$, nie są połączone krawędzią w grafie G.*

Fakt 2.10. *Każde dwa wierzchołki z różnych klik $K_1, K_2 \in Hex(G)$ takie, że $f^{(12)}(K_1) = f^{(12)}(K_2)$, nie mają wspólnego sąsiada w grafie G.*

Wprowadźmy teraz algorytm SelectBroadcastNodes który będzie jednym z kluczowych elementów głównego algorytmu HexagonsBroadcasting. Idea prezentowanego algorytmu polega na wyborze małej liczby wierzchołków biorących udział w rozpowszechnianiu informacji. Wierzchołki te nazwiemy wierzchołkami brzegowymi (ang. broadcast nodes), a ich liczba będzie rzędowo porównywalna z wielkością maksymalnego zbioru niezależnego. Algorytm SelectBroadcastNodes przyporządkowuje każdemu wierzchołkowi dwa zbiory $F_{out}(\cdot)$ i $F_{in}(\cdot)$. Wierzchołkami brzegowymi nazwiemy te wierzchołki dla których zbiory $F_{out}(\cdot)$ i $F_{in}(\cdot)$ są niepuste, przy czym funkcja F_{out} będzie użyta wtedy, gdy $K \in Hex(G)$ wysyła informację do sąsiednich heksagonów podczas gdy funkcja F_{in} będzie użyta do rozpowszechnienia informacji wewnątrz heksagonu K. Przedstawmy teraz pomocniczy algorytm SelectBroadcastNodes który wybiera wierzchołki brzegowe.

Zauważmy że algorytm SelectBroadcastNodes każdej parze heksagonów $\{K, K'\}$ połączonych co najmniej jedną krawędzią przyporządkowuje dokładnie jedną parę powiązanych brzegowych wierzchołków $v_1 \in K$, $v_2 \in K'$ połączonych krawędzią.

Bazując na funkcji $F_{in} : V(G) \to \mathcal{P}(\{1, \ldots, 18\})$ oraz na funkcji $F_{out} : V(G) \to \mathcal{P}(\{1, \ldots, 12\})$ tworzymy odpowiednio zbiory $U_{out}(\cdot, \cdot)$ i $U_{in}(\cdot, \cdot)$ zawierające wierzchołki brzegowe. Zbiory zostaną użyte do konstrukcji struktury rozpowszechniającej informację. Wierzchołki z $U_{out}(i, k)$ wysyłają informację z heksagonu K takiego, że $f^{(12)} = i$, do heksagonu K' o własności $f^{(12)}(K') = k$. Wierzchołki ze zbiorów $U_{in}(i, 1), \ldots, U_{in}(i, 18)$ będą użyte do rozpowszechniania informacji wewnątrz heksagonu K o własności $f^{(7)} = i$. Formalnie rzecz ujmując:

Zauważmy, że dla każdego $1 \leqslant t \leqslant 12$ zbiór $U_{out}(t, t)$ jest pusty, stąd liczba niepustych zbiorów U_{in} i U_{out} nie przekracza $11 \times 12 + 7 \times 18 = 258$.

Poniższy lemat udowadnia poprawność wysyłania komunikatów z kliki $K \in Hex(G)$ do jej sąsiednich heksagonów.

Lemat 2.11. *Niech $1 \leqslant i \leqslant 12$, $1 \leqslant k \leqslant 12$ i $A \subseteq U_{out}(i, k)$. Jeśli $B \subseteq V(G)$ jest zbiorem wierzchołków powiązanych z A i jednocześnie zawartych w heksagonach na których funkcja $f^{(12)}$ jest równa k to wtedy $B \subseteq Inf(A)$.*

Dowód. Załóżmy, że $1 \leqslant i \leqslant 12$, $1 \leqslant k \leqslant 12$. Niech $v_1 \ldots v_t \in K$ będą wierzchołkami z $U_{out}(i, k)$. Zauważmy, że dla każdego wierzchołka v_j zbiór $F_{out}(v_j)$ zawiera

SelectBroadcastNodes

Wejście: Jednostkowy graf dyskowy G

Wyjście: Funkcje $F_{in} : V(G) \to \mathcal{P}(\{1, \ldots, 18\})$, $F_{out} : V(G) \to \mathcal{P}(\{1, \ldots, 12\})$, gdzie $\mathcal{P}(A)$ to zbiór wszystkich podzbiorów zbioru A.

1. Dla każdego $v \in V(G)$ podstaw $F_{in}(v) \equiv \emptyset$ i $F_{out}(v) \equiv \emptyset$.
2. FOR $i = 1$ TO 7 DO: FOR $k = 1$ TO 18 DO:
   Dla każdego $K \in Hex(G)$ takiego, że $f^{(7)}(K) = i$ i $f^{(7)}(N_k(K)) > f^{(7)}(K)$ równolegle wykonaj:
   a) $V_k(K) := \{v \in V(N_k(K)) : N(v) \cap K \neq \emptyset\}$
   b) Jeśli $V_k(K) \neq \emptyset$ wtedy
      i. Wybierz wierzchołek $v_k \in V_k(K)$ i podstaw

$$F_{in}(v_k) := F_{in}(v_k) \cup \{\bar{k}\};$$

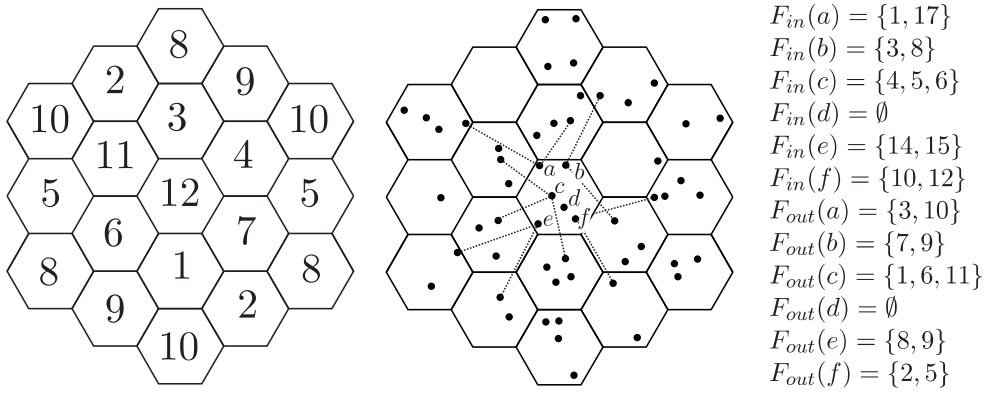
$$F_{out}(v_k) := F_{out}(v_k) \cup \{f^{(12)}(K)\}.$$

      ii. Wybierz wierzchołek $w \in N(v_k) \cap K$ i podstaw:

$$F_{in}(w) := F_{in}(w) \cup \{k\};$$

$$F_{out}(w) := F_{out}(w) \cup \{f^{(12)}(N_k(K))\}.$$

   c) Wierzchołki v_k i w nazywamy *powiązanymi* wierzchołkami brzegowymi.



Rysunek 2.4. Wierzchołki z jednego heksagonu i odpowiednio zbiory $F_{in}(\cdot)$ i $F_{out}(\cdot)$ oraz wierzchołki z sąsiednich heksagonów powiązane z nimi.

k wtedy i tylko wtedy gdy v_j jest powiązany z co najmniej jednym wierzchołkiem z sąsiedniego heksagonu K' takiego, że $f^{(12)}(K') = k$.

Oznaczmy przez W zbiór wierzchołków powiązanych z wierzchołkami $\{v_1 \ldots v_t\}$. Zauważmy, że dwa wierzchołki z W nie mogą znajdować się w tej samej klice. Wy-

BroadcastSets
Wejście: Jednostkowy graf dyskowy G oraz funkcja $F_{in} : V(G) \to \mathcal{P}(\{1, \ldots, 18\})$ i funkcja $F_{out} : V(G) \to \mathcal{P}(\{1, \ldots, 12\})$.
Wyjście: Zbiory $U_{out}(\cdot, \cdot)$ i $U_{in}(\cdot, \cdot)$
1. Niech K_v oznacza klikę $K \in Hex(G)$ taką, że $v \in K$.
2. FOR $i = 1$ TO 12 DO: FOR $k = 1$ TO 12 DO:

$$U_{out}(i, k) := \left\{v \in V : k \in F_{out}(v) \text{ i } f^{(12)}(K_v) = i\right\}$$

3. FOR $i = 1$ TO 7 DO: FOR $k = 1$ TO 18 DO:

$$U_{in}(i, k) := \left\{v \in V : k \in F_{in}(v) \text{ i } f^{(7)}(K_v) = i\right\}$$

nika to z faktu, że do każdej pary heksagonów przypisaliśmy co najwyżej jedną parę powiązanych wierzchołków brzegowych. Łącząc fakt 2.10 z obserwacją, że dla tych różnych heksagonów wartość funkcji $f^{(12)}$ wynosi k wnioskujemy, że żadna para wierzchołków ze zbioru W nie ma wspólnego sąsiada w G. W szczególności nie ma też wspólnego sąsiada w $\{v_1 \ldots v_t\}$. Stąd jeśli $v_1 \ldots v_t$ wysyłają wiadomość jednocześnie w tej samej rundzie, to wierzchołki powiązane z nimi (odnosi się to również do zbioru W) otrzymują tę wiadomość bez interferencji (zakłóceń).

Rozpatrzmy teraz wierzchołki $k_1, k_2 \in U_{out}(i, k)$ zawarte w dwóch heksagonach K_1, K_2. Ponieważ z definicji U_{out} wynika, że $f^{(12)}(K_1) = f^{(12)}(K_2) = i$, dlatego korzystając z faktu 2.10, wierzchołki k_1 i k_2 nie mają wspólnego sąsiada. Reasumując, jeżeli wierzchołki ze zbioru $U_{out}(i, k)$ wysyłają informację w tej samej rundzie, to wierzchołki powiązane z nimi otrzymują ją bez interferencji (zakłóceń). □

Przedstawmy teraz lemat mówiący o poprawności rozpowszechniania informacji wewnątrz danego heksagonu.

Lemat 2.12. *Niech $1 \leqslant i \leqslant 7$, $1 \leqslant k \leqslant 18$ i załóżmy, że wierzchołki ze zbioru $A \subseteq U_{in}(i, k)$ wysyłają informacje w tej samej rundzie. Wtedy dla każdego $K \in Hex(G)$ takiego, że $K \cap A \neq \emptyset$ mamy $K \subseteq Inf(A)$.*

Dowód. Niech $1 \leqslant i \leqslant 7$, $1 \leqslant k \leqslant 18$. Zauważmy, że dla każdego wierzchołka brzegowego v, zbiór $F_{in}(v)$ zawiera k wtedy i tylko wtedy gdy v jest powiązany z wierzchołkiem w należącym do $N_k(K)$. Tak więc zbiór $V(K \cap U_{in}(i, k))$ ma co najwyżej jeden element. Co więcej dwa różne wierzchołki $v, v' \in U_{in}(i, k)$ są zawarte w heksagonach, dla których funkcja $f^{(7)}$ przyjmuje wartość i. Stąd z Faktu 2.9, jeśli wierzchołki $v_1, \ldots v_s \in U_{in}(i, k)$ wysyłają informacje w tej samej rundzie synchro-

nicznej, wtedy wszystkie wierzchołki z heksagonów, w których $v_1, \ldots v_s$ są zawarte, otrzymają te wiadomości bez interferencji. □

Jesteśmy teraz gotowi przedstawić główny algorytm propagacji wiadomości ze źródła $s \in V(G)$ do całej sieci z uwzględnieniem kolizji.

HexagonsBroadcasting
Wejście: Jednostkowy graf dyskowy G i wierzchołek $s \in V(G)$ będący źródłem nadającym informację.
Wyjście: Rozpropagowana informacja w całym grafie G.
1. Uruchom algorytm SelectBroadcastNodes na grafie G.
2. Uruchom algorytm BroadcastSets bazując na wyniku algorytmu SelectBroadcastNodes.
3. Wierzchołek s wysyła informacje do wszystkich swoich sąsiadów.
4. FOR $t = 1$ TO R DO:
   a) FOR $i = 1$ TO 12 DO: FOR $k = 1$ TO 12 DO: IF $k \neq i$ DO:
      - Wierzchołki z $U_{out}(i, k)$ które otrzymały informacje, równolegle wysyłają ją do wszystkich sąsiadów
   b) FOR $i = 1$ TO 7 DO: FOR $k = 1$ TO 18 DO:
      - Wierzchołki z $U_{in}(i, k)$ które otrzymały informacje, równolegle wysyłają ją do wszystkich sąsiadów

---

Zauważmy, że mając dany algorytm HexagonsBroadcasting działający na źródle s możemy zdefiniować strukturę rozpowszechniania informacji $U_1, U_2, U_3, \ldots$ o własnościach podanych w definicji 2.8. W tym wypadku za zbiór U_i podstawiamy zbiór wierzchołków który w i-tej rundzie algorytmu HexagonsBroadcasting rozsyła informacje.

Z drugiej strony krok 1 i krok 2 algorytmu HexagonsBroadcasting konstruują całą strukturę potrzebną do rozpowszechniania informacji. Ponieważ algorytm SelectBroadcastNodes działa w stałym czasie, jak również konstrukcja zbiorów U_{out} i U_{in} zdefiniowanych przez algorytm BroadcastSets zajmuje stałą liczbę synchronicznych rund więc wszystkie wstępne obliczenia potrzebne do odświeżania struktury rozpowszechniania informacji odbywają się w stałym czasie.

Twierdzenie 2.13. *Niech R będzie promieniem grafu G o źródle w wierzchołku s. Latencja algorytmu* HexagonsBroadcasting *(czyli liczba synchronicznych rund po których wszystkie wierzchołki otrzymają informację) wynosi co najwyżej* $258R + 1$.

Dowód. Niech $U_1, U_2, U_3, \ldots$ będzie strukturą rozpowszechniania informacji wygenerowaną przez algorytm HexagonsBroadcasting i niech $K \in Hex(G)$ będzie heksagonem takim, że $s \in K$. W pierwszym kroku s wysyła informację do całego heksagonu K (czyli $K \subseteq Inf(U_1)$). Z lematu 2.11 wynika, że w następnych $11 \times 12 = 132$ krokach wierzchołki brzegowe zawarte w klice K wysyłają informację do wszystkich wierzchołków powiązanych z nimi i znajdujących się w $N(K)$. Z lematu 2.12 wiemy, że w następnych 7×18 krokach wierzchołki brzegowe rozpowszechnią informację wewnątrz heksagonów należących do $N(K)$. Stąd uzyskujemy, że $\bigcup_{l=1}^{258+1} Inf(U_l)$ zawiera wszystkie wierzchołki z K jak również wszystkie wierzchołki z $N(K)$. Tak więc wszyscy sąsiedzi wierzchołka s w grafie G otrzymują od niego informację w pierwszych $258+1$ krokach. Możemy teraz powtórzyć rozumowanie zamieniając wierzchołki brzegowe z heksagonu K na wierzchołki brzegowe zawarte w heksagonach $N(K)$. Stąd po kolejnych 258 krokach wszystkie wierzchołki będące w odległości 2 od s otrzymają informację (czyli wszystkie wierzchołki odległe od s o co najwyżej 2 są zawarte w $\bigcup_{l=1}^{2\times 258+1} Inf(U_l)$). Analogicznie wszystkie wierzchołki odległe od s o co najwyżej R, są zawarte w zbiorze $\bigcup_{l=1}^{258R+1} Inf(U_l)$. □

2.4. Uogólnienia problemu rozpowszechniania informacji

W niniejszym paragrafie opiszemy w jaki sposób przedstawione techniki podziału na heksagony i wyboru powiązanych wierzchołków można zastosować można do szerszej klasy problemów, związanych z rozpowszechnianiem informacji w jednostkowych grafach dyskowych.

Wpierw pokażemy, że algorytm HexagonsBroadcasting może być dostosowany do dynamicznie zmieniającej się sieci modelowanych przez UDG. Podkreślmy, iż wszystkie dotychczasowe algorytmy rozpowszechniania informacji nie posiadały tej własności. Przez zmiany w sieci rozumieć będziemy trzy następujące zdarzenia:

1. Dodanie wierzchołka v takiego, że $G \cup \{v\}$ jest spójny.
2. Usunięcie wierzchołka $v \in V(G) \setminus \{s\}$ w ten sposób, by $G \setminus \{v\}$ pozostał spójny.
3. Zmiana położenia wierzchołka $v \in V(G)$ taka, że po dokonaniu tej operacji graf G pozostaje nadal spójny.

Niech s będzie źródłem wiadomości i niech a będzie pewną z góry ustaloną stałą. Poniższy algorytm rozpowszechnia informację w dynamicznie zmieniającej się sieci.

Ponieważ procedury SelectBroadcastNodes oraz BroadcastSets zajmują stałą liczbę rund, więc przy ich pomocy można w stałej liczbie rund odświeżyć strukturę rozpowszechniania informacji za pomocą algorytmu DynamicBroadcasting. Stąd widać, że DynamicBroadcasting jest efektywnym narzędziem

DynamicBroadcasting
1. Uruchom algorytm HexagonsBroadcasting.
2. Przerwij krok 4 algorytmu HexagonsBroadcasting po a iteracjach.
3. Wróć do kroku 1.

do propagacji informacji w dynamicznych sieciach. Zauważmy, że jeśli od pewnej rundy zmiany w sieci (takie jak dodanie, przesunięcie bądź usunięcie wierzchołka) przestaną zachodzić, wtedy w $258R + (1/a)R + O(a)$ kolejnych rundach wszystkie wierzchołki otrzymają informacje ze źródła. Oczywiście w praktycznych implementacjach tego algorytmu informacja rozpowszechniana jest znacznie szybciej i okazuje się, że rozprzestrzenia się na całą sieć nawet w trakcie zachodzenia zmian.

We wszystkich wcześniejszych algorytmach, które jak wspominaliśmy we wstępie bazują na konstrukcji drzewa BFS, zbudowanie struktury rozpowszechniającej informacje zajmuje czas $O(R)$. Stąd nie da się ich efektywnie zmodyfikować tak aby uzyskać algorytm analogiczny do DynamicBroadcasting. W przypadku dynamicznej modyfikacji wcześniejszych algorytmów, czyli za pomocą parametru a do odświeżania struktury, jeśli w pewnej rundzie synchronicznej zmiany przestaną zachodzić, wtedy dopiero po $O(R^2/a + a)$ rundach mamy pewność, że informacja została rozpropagowana w całej sieci.

Przejdźmy teraz do kolejnego uogólnienia problemu rozpowszechniania informacji, a mianowicie zagadnienia plotkowanie (ang. gossiping). W problemie plotkowania każdy wierzchołek posiada unikalną informację. Celem jest rozpowszechnienie wszystkich informacji do wszystkich wierzchołków w sieci. Oczywiście zakładamy że podczas równoczesnego przesyłania informacji zachodzić mogą interferencje.

Przy problemie plotkowania rozważa się dwa modele przesyłu informacji biorąc pod uwagę to, czy dwie wiadomości można łączyć i przesyłać jako jedną - większą, czy też wielkość przesyłanych informacji jest ograniczona. Pierwszy przypadek nazywamy modelem długich komunikatów (ang. unbounded-size message model), a drugi nazywamy modelem krótkich komunikatów (ang. unit-size message model). W przypadku krótkich komunikatów ograniczeniem dolnym na latencję plotkowania jest $V(G) + D - 1$, gdzie D jest średnicą grafu G. Podczas gdy przy długich komunikatach ograniczeniem dolnym na latencję plotkowania jest $\Delta(G) + D - 1$ ($\Delta(G)$ jest maksymalnym stopniem grafu G) W zagadnieniu plotkowania, podobnie jak w problemie rozpowszechniania informacji, ilość synchronicznych rund potrzebna by wszystkie wiadomości dotarły do wszystkich wierzchołków nazywana jest latencją.

Dla dowolnych grafów problem plotkowania, przy założeniu długich komunikatów, był rozpatrywany w pracach [12], [13], [14], [29] i [30]. Współczynniki aproksymacji algorytmów zawartych w tych pracach są rzędowo większe niż stała. Przy tym samym założeniu, dla jednostkowych grafów dyskowych Gandhi, Parthasarathy i Mishra w pracy [27] zaprezentowali stałą aproksymację optymalnego rozwiązania problemu plotkowania. Współczynnik latencji z tej pracy można oszacować przez liczbę $1944(D + |V(G)|)$. Najlepszy algorytm rozwiązujący problem plotkowania w modelu krótkich komunikatów dla UDG został zaprezentowany w pracy [34]. Huang, Scott, Du, Hongwei i Park pokazali w niej algorytm plotkowania o latencji $27(|V(G)| + D - 1)$.

Poniżej zaprezentujemy algorytm GossipingInUDG który rozwiązuje problem plotkowania w jednostkowych grafach dyskowych w modelu z długimi komunikatami o latencji $7\Delta + 258D$, poprawiając w ten sposób wynik z pracy [27]. Co więcej, jest to pierwszy algorytm który buduje strukturę plotkowania w czasie stałym, nie korzystając z drzewa BFS. Dlatego, podobnie jak miało to miejsce w algorytmie DynamicBroadcasting, istnieje możliwość zaadoptowania naszego algorytmu do dynamicznych sieci, w których zachodzą dodawanie, usuwanie i przemieszczanie wierzchołków.

GossipingInUDG

1. Znajdź funkcje F_{out} i F_{in} używając algorytmu SelectBroadcastNodes.
2. Dla każdego $K \in Hex(G)$ i wierzchołków $v_1, v_2, v_3 \ldots = V(K)$ zdefiniujmy

$$c(v_i) = 7i + f^{(7)}(K) \text{ dla każdego } 1 \leqslant i \leqslant |V(K)|.$$

3. Niech v będzie wierzchołkiem brzegowym. Zdefiniujmy

$$c_2(v) = \max\{c(w) : \exists_u u \in N(v) \text{ i } w \in N(u)\}$$

   ($c_2(v)$ oznacza maksymalną wartość funkcji c w 2-sąsiedztwie wierzchołka v).
4. Ustaw stan wszystkich wierzchołków brzegowe (czyli wierzchołki, które mają $F_{out} \neq \emptyset$ i $F_{in} \neq \emptyset$) na nieaktywny.
5. Równolegle wykonaj:
   a) Każdy nieaktywny wierzchołek v wysyła swoją wiadomość w $c(v)$-tej synchronicznej rundzie.
   b) Każdy wierzchołek brzegowy v' jest aktywowany w $c_2(v')+1$ synchronicznej rundzie, po czym wysyła konkatenację otrzymanych wiadomości w rundach zdefiniowanych w punkcie 4 algorytmu HexagonsBroadcasting.

Twierdzenie 2.14. *Algorytm* GossipingInUDG *dostarcza wszystkie początkowe informacje do wszystkich wierzchołków w co najwyżej* $7\Delta + 258D$ *synchronicznych rundach.*

Dowód. Niech $v \in V(K)$ i $v' \in V(K')$ będą dwoma różnymi wierzchołkami takimi, że $c(v) = c(v')$. Z definicji funkcji c mamy, że $f^{(7)}(K) = f^{(7)}(K')$ i $K \neq K'$. Stąd bezpośrednio z faktu 2.9 wiemy, że w kroku 5a żaden wierzchołek z K nie otrzyma wiadomości z v', dopóki wszystkie wierzchołki z K nie otrzymają wiadomości z v. W kroku 5b wierzchołki brzegowe wysyłają informację po tym, jak wszystkie wierzchołki z ich 2-sąsiedztwa wysłały początkową informację. Stąd nie ma interferencji wytwarzanej poprzez wierzchołki brzegowe i wierzchołki wysyłające początkową informację.

Zauważmy, że używamy co najwyżej 7Δ rund synchronicznych, by wysłać początkowe wiadomości w kroku 5a (wynika to z tego, że mamy 7 klas heksagonów i wierzchołki w heksagonie tworzą klikę). Liczba rund synchronicznych zawartych w kroku 5b jest równa latencji struktury rozpowszechniającej informację zdefiniowanej w algorytmie HexagonsBroadcasting. Dokładniej, po tym jak wierzchołki z 2-sąsiedztwa wyślą swoją początkową wiadomość, wszystkie wierzchołki brzegowe wysyłają informacje zgodnie z algorytmem HexagonsBroadcasting. Latencja dalszego przesyłu wynika wprost z twierdzenia 2.13. Stąd krok 5b zajmuje co najwyżej $258D$ rund synchronicznych.

Podsumowując algorytmGossipingInUDG ma latencję co najwyżej $7\Delta + 258D$. □

Kolejnym uogólnieniem problemu rozpowszechniania informacji jest zagadnienie w którym źródło informacji wielokrotnie wysyła wiadomości do sieci. Zagadnienie to rozpatrzyli Gandhi, Parthasarathy i Mishra w pracy [27] i nazwali rozpowszechnianiem wielu wiadomości (ang. single sources multiple messages broadcasting). Autorzy założyli, że źródło może wielokrotnie wysyłać wiadomości w różnych odstępach czasu i zdefiniowali latencję jako czas, który jest potrzebny by wszystkie wiadomości ze źródła dotarły do wszystkich wierzchołków w sieci. Niech M będzie liczbą wiadomości, które mają zostać wysłane. W pracy [27] zaprezentowany algorytm rozpowszechnienia wielu wiadomości ma latencję co najwyżej $194(M-1) + 7128(R-2)$.

Poniżej zaprezentujemy modyfikację algorytmu HexagonsBroadcasting dający w wyniku strukturę rozpowszechnienia wielu wiadomości o latencji $518M + 258R$. Zmiany obejmować będą tylko punkty 3 i 4 algorytmu HexagonsBroadcasting i będą następujące:

MultipleMessagesBroadcasting *Wejście:* Jednostkowy graf dyskowy G i wierzchołek $s \in V(G)$. Wierzchołek s jest źródłem posiadajacym M informacji. *Wyjście:* M rozpropagowanych informacja w całym grafie G.

1. Uruchom algorytm SelectBroadcastNodes na grafie G.
2. Uruchom algorytm BroadcastSets bazując na wyniku algorytmu SelectBroadcastNodes.
3. Powtórz $M + \lceil R/2 \rceil$ razy:
   (a) Jeśli s zawiera nową wiadomość to roześlij ją do wszystkich sąsiadów
   (b) Powtórz dwa razy:
   — FOR $i = 1$ TO 12 DO: FOR $k = 1$ TO 12 DO: IF $k \neq i$
   - wierzchołki z $U_{out}(i, k)$ które otrzymały informacje której jeszcze nie wysłały wcześniej, wysyłają ją.
   — FOR $i = 1$ TO 7 DO: FOR $k = 1$ TO 18 DO:
   - Wierzchołki z $U_{in}(i, k)$ które otrzymały informacje której jeszcze nie wysłały wcześniej, wysyłają ją.

Twierdzenie 2.15. *Jeśli w zmodyfikowanym algorytmie* HexagonsBroadcasting *źródło wysyła wiadomość co* 259 *synchronicznych rund, wtedy każdy wierzchołek w sieci otrzyma wszystkie wiadomości po czasie co najwyżej* $518M + 258R$*, gdzie* M *jest liczbą wiadomości.*

Dowód. Podzielmy strukturę rozpowszechniania informacji wygenerowaną przez HexagonsBroadcasting na przedziały wielkości 258 rund (pomijając pierwszą rundę). Zauważmy, że jeśli wierzchołek brzegowy otrzyma wiadomość w t-tym przedziale, wtedy w $(t + 1)$-szym przedziale informacja ta zostanie rozpowszechniona do wszystkich jego sąsiadów. Stąd wierzchołek ten może wysłać informację w $(t + 2)$-gim przedziale. Fakt, że nie zajdą interferencje, wynika bezpośrednio z dowodu twierdzenia 2.13. Stąd posłużenie się dwoma przedziałami czasowymi długości 258 plus jedną rundą (potrzebną by s wysłał nową wiadomość) wystarczy do tego, by wszystkie M wiadomości zostało dostarczonych do wszystkich wierzchołków w sieci. □

W klasycznym problemie rozpowszechniania informacji wszystkie wierzchołki w sieci, by się komunikować, używają tej samej częstotliwości radiowej.Dlatego, gdy dwa wierzchołki jednocześnie nadają na obszarze wspólnego zasięgu ich nadajników, zachodzi interferencja. Jednakże w rzeczywistych sieciach dostępna jest więcej niż jedna częstotliwość komunikacji. W przypadku, gdy do dyspozycji jest kilka różnych kanałów komunikacji, to informacje wysłane na różnych częstotliwościach od dwóch różnych sąsiadów zostaną odebrane bez kolizji. Celem wielokanałowe-

go rozpowszechniania informacji jest jak najszybsze rozpowszechnienie informacji przy użyciu jak najmniejszej liczby kanałów komunikacyjnych.

Poniżej opiszemy jak wykorzystując algorytm HexagonsBroadcasting można przyporządkować odpowiednim grupom wierzchołków kanały, tak aby używając 132 częstotliwości algorytm wielokanałowego rozpowszechniania informacji miał latencję $2R$. Zauważmy, że jeśli dla dowolnej pary indeksów $(i_1, k_1) \neq (i_2, k_2)$, wierzchołki ze zbiorów $U_{out}(i_1, k_1)$ i $U_{out}(i_2, k_2)$ używają różnych kanałów, wtedy mogą wysyłać posiadane przez nie informacje jednocześnie i nie zachodzą interferencje. Ten sam argument działa dla $U_{in}(i_1, k_1)$ i $U_{in}(i_2, k_2)$ jeśli $((i_1, k_1) \neq (i_2, k_2))$. Stąd jeśli przypiszemy 132 różnych częstotliwości ($11 \times 12 \leqslant 132$, $7 \times 18 \leqslant 132$) do każdego zbioru typu U_{out} oraz 126 różnych kanałów zbiorom U_{in}, wtedy będziemy mogli zmodyfikować algorytm HexagonsBroadcasting tak aby korzystał on z 132 kanałów komunikacji mając jednocześnie latencję $2R$.

3. Algorytmy klastrujące

3.1. Wprowadzenie

Jedną z heurystyk często wykorzystywanych do rozwiązywania aproksymacyjnego problemów w modelu rozproszonym jest tak zwane klastrowanie. Procedura ta polega na podziale grafu w sposób rozproszony na spójne podgrafy, tak zwane klastry, w celu ich późniejszego wykorzystania. Dokładnie rzecz ujmując, jeśli podział ma odpowiednie własności, to dany problem może być rozwiązywany lokalnie wewnątrz każdego z klastrów a takie lokalne rozwiązania dają w sumie dobrą aproksymację rozwiązania globalnego. Przedstawione powyżej podejście pozwala na rozwiązanie wielu istotnych problemów w szybkim czasie. Między innymi, dzięki technice klastrowania, można uzyskać bardzo dobrą aproksymacją takich struktur jak maksymalny zbiór dominujący, największy zbiór niezależny, pakowanie itp.

Niniejszy rozdział poświęcony jest przedstawieniu dwóch algorytmów konstruujących podziały grafu dyskowego. Pierwszy z algorytmów zaprezentowany jest w podrozdziale 3.2. Tworzy on podział na klastry, które zawierają stosunkowo dużą liczbę wierzchołków oraz posiadają na tyle małą średnicę, aby zapewnić efektywną komunikację wewnątrz podgrafów podziału. Dzięki wyżej wymienionym własnościom przedstawiony podział na klastry może być zastosowany do konstrukcji aproksymacji najmniejszego zbioru k-dominującego, największego skojarzenia i najmniejszego spójnego zbioru dominującego. W drugiej części rozdziału (w podrozdziale 3.4) zaprezentowany jest algorytm tworzący podział na klastry, których wielkość mieści się w z góry zadanym przedziale. Przedstawiona konstrukcja przydatna jest w upraszczaniu protokołów komunikacyjnych oraz rozproszonym zbieraniu i przetwarzaniu danych.

Część z prezentowanych rezultatów pochodzi z prac [42] i [43]

3.2. Silny zbiór

Jak już wspomniane zostało we wprowadzeniu, rozpatrywanym przez nas problem polega na znalezieniu podziału grafu na „duże” składowe o „małej” średnicy,

nazwanym *zbalansowanym* klastrowaniem. Poniżej powiemy jak skonstruować taki podział wybierając najpierw odpowiedni podzbiór wierzchołków, tak zwanych „liderów", a następnie na ich podstawie wyznaczając podział na klastry.

Zauważmy, że dla danego zbioru liderów L składającego się z l wierzchołków grafu G można zdefiniować podział $K = \{K_1, K_2, \ldots, K_l\}$, w ten sposób, że wierzchołek $v \in V(G)$ należy do i-tego klastra ($v \in K_i$) wtedy i tylko wtedy gdy i–ty lider jest najbliższy v spośród wszystkich liderów. Jeśli więcej niż jeden lider jest najbliższy v, to v będzie przyłączany tego lidera o najmniejszym identyfikatorze ID. W klasycznej już pracy [3] Awerbuch, Goldberg, Luby i Plotkin do konstrukcji podziału wykorzystali tak zwany silny zbiór (ang. ruling set) jako zbiór „liderów". Na potrzeby tego rozdziału przyjmujemy następującą definicję zbioru (d, b)–silnego:

Definicja 3.1. *Niech d i b będą ustalonymi liczbami naturalnymi i niech $dist(u, u')$ będzie długością najkrótszej ścieżki pomiędzy u a u' w grafie G. Podzbiór U zbioru wierzchołków $V(G)$ nazywamy zbiorem (d, b)–silnym, jeżeli spełnia następujące własności:*

(a) Jeżeli $u, u' \in U$, to $dist(u, u') \geqslant d$.
(b) Zbiór U jest b–dominujący, to znaczy dla każdego $v \in V \setminus U$ istnieje wierzchołek $u \in U$ taki, że $dist(v, u) \leqslant b$.

We wspomnianej pracy [3] Awerbuch, Goldberg, Luby i Plotkin przedstawili działający dla dowolnego grafu G w czasie $O(d \log |V(G)|)$ rund synchronicznych algorytm rozproszony, który dla ustalonego parametru d znajduje $(d, d \log |V(G)|)$–silny zbiór. Zbiór taki wyznacza podział na grafy o średnicy co najwyżej $2d \log |V(G)|$, czyli korzystając z algorytmu rozproszonego przeszukiwania wszerz BreadthFirstSearch, „liderzy" szybko mogą poznać strukturę swoich klastrów. W dodatku dowolni dwaj „liderzy" są w odległości co najwyżej d, w związku z tym grafy podziału są dość duże, aby podział mógł być wykorzystany w zastosowaniach. Jak widać właściwy zbiór liderów, który jest zbiorem silnym, zapewnia podział o własnościach, które są użyteczne do konstrukcji aproksymacji rozwiązań. Prace nad silnymi zbiorami w ogólnych grafach kontynuowane były między innymi w pracach [52] oraz [56].

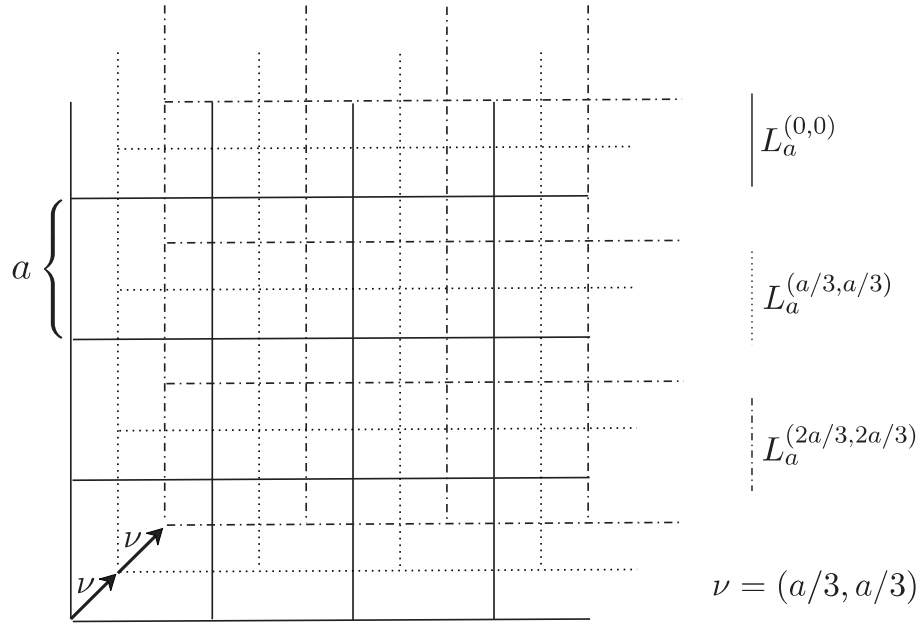
W niniejszym rozdziale rozpatrywany jest problem znalezienia zbioru silnego w jednostkowych grafach dyskowych. Zaprezentowany jest deterministyczny algorytm rozproszony, który w modelu LOCAL+COORD, dla danego parametru d, znajduje zbiór $(d + 1, 3d)$–silny, a co za tym idzie wyznacza zbalansowane klastrowanie. Czas działania tego algorytmu to $O(poly(d))$ rund synchronicznych. Oznacza to, że dla rozpatrywanej klasy grafów poprawimy zarówno współczynnik dominowania jak i złożoność czasową algorytmu z pracy [3].

Rozdział ten jest zorganizowany w następujący sposób. Najpierw zaprezentowana jest główna idea algorytmu RulingSet a następnie przedstawione są dwa algorytmy RulingSetInSquare oraz DeletingLocalCollisions, które formalizują ideę znajdowania lokalnych wyników i usuwania kolizji. Główne twierdzenie 3.2 mówi o poprawności głównego algorytmu RulingSet. Następnie pokazane są zastosowania prezentowanego algorytmu do znalezienia $O(d)$ aproksymacji zbioru d-dominującego w czasie $O(poly(d))$ rund synchronicznych. Na zakończenie udowodnione jest, że przedstawiona konstrukcja $(d+1, 3d)$–silnego zbioru polepsza złożoność czasową algorytmów z pracy [16] znajdujących maksymalne skojarzenie i minimalny spójny zbiór dominujący.

Kluczową ideą algorytmu RulingSet jest metoda trzech siatek (ang. three lattices method), która w rozdziale 2.2 została użyta do rozwiązania problemu konstrukcji $(1 + O(1/d))$ aproksymacji minimalnego drzewa rozpinającego . Przypomnijmy, że w tej metodzie rozpatrujemy trzy różne siatki: $L_a^{(0,0)}$, $L_a^{(a/3,a/3)}$ oraz $L_a^{(2a/3,2a/3)}$ zawierające poziome i pionowe linie odległe od siebie o a oraz przechodzące przez punkty $(0, 0)$, $(a/3, a/3)$ in $(2a/3, 2a/3)$, odpowiednio. Zbiór kwadratów ograniczonych prostymi należącymi do siatki $L_a^{(\nu_1,\nu_2)}$ oznaczamy przez $\mathcal{S}_a^{(\nu_1,\nu_2)}$. Oznaczamy też przez $G[\mathcal{S}]$ podgraf grafu G indukowany na wierzchołkach zawartych w kwadracie $\mathcal{S}$ (czyli wierzchołki ze zbioru $\{v \in V(G) : v \in \mathcal{S}\}$). W pierwszym kroku algorytmu obliczmy optymalne rozwiązania wewnątrz wszystkich $G[\mathcal{S}]$ takich, że $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$. W kolejnych dwóch krokach poprawiamy rozwiązania wewnątrz podgrafów indukowanych przez kwadraty z $\mathcal{S}_a^{(a/3,a/3)}$ i $\mathcal{S}_a^{(2a/3,2a/3)}$ kolejno.

Formalną implementacją pierwszego kroku głównego algorytmu jest algorytm RulingSetInSquare. Oblicza on lokalnie optymalne rozwiązanie w grafach indukowanych przez kwadraty z $\mathcal{S}_a^{(0,0)}$. Przy pomocy tego algorytmu można znaleźć zbiór $(d + 1, d)$–silny w każdym $G[\mathcal{S}]$, $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$, w czasie $O(d^3)$. Ponieważ zbiory $(d + 1, d)$–silne są konstruowane niezależnie dla każdego $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$, więc mogą istnieć pary wierzchołków ze zbiorów wyznaczonych w różnych kwadratach, które znajdują się w odległości mniejszej niż d. Aby wyeliminować takie pary użyty jest algorytm DeletingLocalCollisions. Czas działania tego algorytmu wynosi $O(d^3)$ synchronicznych rund. Wykorzystany jest on dwukrotnie. Po raz pierwszy działa wewnątrz grafów indukowanych przez kwadraty z $\mathcal{S}_a^{(a/3,a/3)}$ i konstruuje zbiór $(d + 1, 2d)$–silny w każdym z nich. Po raz drugi użyty jest on do konstrukcji zbiorów silnych wewnątrz kwadratów z $\mathcal{S}_a^{(2a/3,2a/3)}$. Ostatecznie wykorzystanie procedury DeletingLocalCollisions umożliwia konstrukcję zbioru $(d + 1, 3d)$–silnego w całym grafie.

Niech $\mathcal{S}$ będzie dowolnym kwadratem o boku a. Przypomnijmy, że przez $G[\mathcal{S}]$ oznaczamy podgraf grafu G indukowany przez wierzchołki $\{v \in V(G) : v \in \mathcal{S}\}$. Algorytm RulingSet działa w następujący sposób.



Rysunek 3.1. Idea metody trzech siatek

RulingSet
Input: Jednostkowy graf dyskowy G oraz ustalone d i a.
Output: $(d+1, 3d)$–silny zbiór grafu G.
(1) Równolegle znajdź zbiór $(d+1, d)$–silny we wszystkich składowych spójności $G[\mathcal{S}]$, dla wszystkich $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$, korzystając z algorytmu RulingSetInSquare. Oznacz rodzinę wynikowych zbiorów przez R.
(2) Uruchom równolegle algorytm DeletingLocalCollisions na wszystkich składowych spójności grafu $R[\mathcal{S}]$ dla każdego $\mathcal{S} \in \mathcal{S}_a^{(a/3,a/3)}$. Oznacz rodzinę wynikowych zbiorów przez R'.
(3) Uruchom równolegle algorytm DeletingLocalCollisions na wszystkich składowych spójności grafu $R[\mathcal{S}]$ dla każdego $\mathcal{S} \in \mathcal{S}_a^{(2a/3,2a/3)}$. Oznacz rodzinę wynikowych zbiorów przez R''.
(4) Zwróć R''.

Twierdzenie 3.2. *Niech G będzie jednostkowym grafem dyskowym i niech $a = 3d+3$. Algorytm* RulingSet *znajduje $(d+1, 3d)$–silny zbiór w grafie G w czasie $O(d^3)$ synchronicznych rund w modelu obliczeń LOCAL+COORD używając tylko krótkich komunikatów.*

Przedstawione poniżej obserwacje są kluczowe w dowodzie głównego twierdzenia.

Lemat 3.3. *Dla dowolnych trzech punktów* $q_0 \in L_a^{(0,0)}$, $q_1 \in L_a^{(a/3,a/3)}$ *oraz* $q_2 \in L_a^{(2a/3,2a/3)}$ *mamy* $\|q_0, q_1\| + \|q_0, q_2\| \geqslant a/3$.

Dowód. Dowód tego lematu znaleźć można w rozdziale 2.2 w którym rozważaliśmy problem minimalnego drzewa rozpinającego. □

Następny lemat wskazuje ograniczenie na moc największego zbioru niezależnego w grafie $G[\mathcal{S}]$. Oznaczmy największy zbiór niezależny w $G[\mathcal{S}]$ przez $\mathrm{MIS}(G[\mathcal{S}])$ a średnicę grafu $G[\mathcal{S}]$ oznaczmy przez $diam(G[\mathcal{S}])$.

Lemat 3.4. *Jeżeli* $\mathcal{S}$ *jest kwadratem o boku* a, *to* $|MIS(G[\mathcal{S}])| \leqslant \frac{4(a+1)^2}{\pi}$. *Co więcej jeżeli* $G[\mathcal{S}]$ *jest spójny, to* $diam(G[\mathcal{S}]) \leqslant \frac{8(a+1)^2}{\pi}$.

Dowód. Oznaczmy przez $C_r(v)$ koło o promieniu r i środku w wierzchołku v (czyli w punkcie na płaszczyźnie odpowiadającemu temu wierzchołkowi). Dla dowolnych dwóch wierzchołków $v_i, v_j \in \mathrm{MIS}(G[\mathcal{S}])$, zachodzi $\|v_i, v_j\| > 1$, tak więc koła $C_{0.5}(v_i)$ oraz $C_{0.5}(v_j)$ są rozłączne. Oczywiście każde $C_{0.5}(v_i)$ leży w kwadracie o boku $a+1$. Ale taki kwadrat może zawierać co najwyżej $\frac{(a+1)^2}{\pi/4}$ rozłącznych kół o promieniu 0.5. Stąd moc $\mathrm{MIS}(G[\mathcal{S}])$ wynosi co najwyżej $\frac{4(a+1)^2}{\pi}$.

Niech $G[\mathcal{S}]$ będzie grafem spójnym oraz $v, w \in V(G[\mathcal{S}])$ będą dwoma wierzchołkami takimi, że najkrótsza ścieżka $p_{v,w}$ pomiędzy v i w ma długość równą średnicy grafu $G[\mathcal{S}]$. Zbiór składający się z co drugiego wierzchołka na ścieżce jest zbiorem niezależnym. Stąd mamy, że $diam(G) \leqslant 2|\mathrm{MIS}(G[\mathcal{S}])| \leqslant \frac{8(a+1)^2}{\pi}$. □

Niech $N_k(u)$ oznacza podgraf G indukowany przez wierzchołki odległe o co najwyżej k od wierzchołka u. Wprowadźmy oznaczenie $\tilde{N}_{v_i} := N_{\lfloor (d-1)/2 \rfloor}(v_i)$.

Lemat 3.5. *Niech* G *będzie grafem spójnym,* $b, d \geqslant 2$ *oraz* U *będzie zbiorem* $(d+1, b)$*–silnym w* G. *Jeżeli* $v_i \neq v_j$ *i* $v_i, v_j \in U$, *to nie ma krawędzi pomiędzy* $\tilde{N}_{v_i}$ *a* $\tilde{N}_{v_j}$. *Jeśli w dodatku* $|U| > 1$, *to dla każdego* $v_i \in U$ *mamy*

$$diam(\tilde{N}_{v_i}) \geqslant \lfloor (d-1)/2 \rfloor \quad i \quad |MIS(\tilde{N}_{v_i})| \geqslant \left\lfloor \frac{1}{2} \lfloor (d-1)/2 \rfloor \right\rfloor + 1.$$

Dowód. Jeśli istniałaby krawędź pomiędzy $\tilde{N}_{v_i}$ a $\tilde{N}_{v_j}$, to istniałaby ścieżka o długości mniejszej niż $2\lfloor (d-1)/2 \rfloor + 1 \leqslant d$ pomiędzy v_i i v_j, co byłoby w sprzeczności z założeniem, że U jest $(d+1, b)$–silnym zbiorem. Ponieważ G jest spójnym grafem i $|U| > 1$ to musi istnieć ścieżka pomiędzy v_i a v_j. Przypuśćmy, że p_{v_i,v_j} jest najkrótszą ścieżką pomiędzy v_i a v_j. Jeśli weźmiemy pierwsze $\lfloor (d-1)/2 \rfloor$ krawędzi na p_{v_i,v_j} to należą one do $\tilde{N}_{v_i}$. Stąd $diam(\tilde{N}_{v_i}) \geqslant \lfloor (d-1)/2 \rfloor$. W dodatku, jeżeli jednostkowy graf dyskowy G ma średnicę k, to $|\mathrm{MIS}(G)| \geqslant \lfloor k/2 \rfloor + 1$, zatem $|MIS(\tilde{N}_{v_i})| \geqslant \left\lfloor \frac{1}{2} \lfloor (d-1)/2 \rfloor \right\rfloor + 1$. □

Wprowadźmy teraz algorytm RulingSetInSquare znajdujący $(d+1, d)$–silny zbiór w grafie $G[\mathcal{S}]$.

RulingSetInSquare
Wejście: Spójny jednostkowy graf dyskowy G, kwadrat $\mathcal{S}$ o boku a oraz ustalona stała d.
Wyjście: zbiór $(d+1, d)$–silny w grafie $G[\mathcal{S}]$.
(1) Niech $M := \emptyset$ będzie wynikowym zbiorem i $Z := \emptyset$ oznacza zbiór wierzchołków które są d–zdominowane przez wierzchołki z M.
(2) Powtarzaj $t = \left\lceil \frac{16(a+1)^2}{\pi d} \right\rceil$ razy:
   (a) Wybierz wierzchołek w z $V(G[\mathcal{S}]) \setminus (M \cup Z)$ o najmniejszym identyfikatorze ID i podstaw $M := M \cup w$.
   (b) Dodaj do zbioru Z wszystkie wierzchołki $u \in V(G[\mathcal{S}])$, dla których $dist(w, u) \leqslant d$.
(3) Zwróć M jako wynik.

Lemat 3.6. *Algorytm* RulingSetInSquare *znajduje zbiór $(d+1, d)$–silny w $G[\mathcal{S}]$ w $O(a^4/d)$ rundach synchronicznych.*

Dowód. Algorytm RulingSetInSquare konstruuje w sposób zachłanny zbiór M, w którym wszystkie wierzchołki są odległe od siebie o co najmniej $d+1$. W dodatku każdy wierzchołek z Z jest d–zdominowany przez wierzchołki z M. Zbiór M nie jest zbiorem $(d+1, d)$–silnym tylko w przypadku, gdy po $t = \left\lceil \frac{16(a+1)^2}{\pi d} \right\rceil$ iteracjach istnieją wierzchołki grafu $G[\mathcal{S}]$ poza zbiorem Z, czyli istnieje zbiór $(d+1, d)$–silny o mocy większej niż t (można M w sposób zachłanny uzupełnić do zbioru $(d+1, d)$–silnego). Wystarczy zatem wykazać, że moc dowolnego zbioru $(d+1, d)$–silnego w $G[\mathcal{S}]$ jest ograniczona z góry przez $t = \left\lceil \frac{16(a+1)^2}{\pi d} \right\rceil$. Niech U będzie dowolnym zbiorem $(d+1, d)$–silnym w $G[\mathcal{S}]$. Przypomnijmy, że przez $\tilde{N}_{v_i}$ oznaczamy podgraf indukowany przez wierzchołki odległe o co najwyżej $\lfloor (d-1)/2 \rfloor$ od v_i w grafie $G[\mathcal{S}]$. Z lematu 3.5 wynika, że nie ma krawędzi pomiędzy $\tilde{N}_{v_i}$ a $\tilde{N}_{v_j}$ gdzie $v_i, v_j \in U$, $v_i \neq v_j$. Stąd $MIS(\tilde{N}_{v_i})$ i $MIS(\tilde{N}_{v_j})$ są zbiorami rozłącznymi i nie ma krawędzi pomiędzy $MIS(\tilde{N}_{v_i})$ i $MIS(\tilde{N}_{v_j})$. Z lematu 3.5 wynika również, że $MIS(\tilde{N}_{v_i}) \geqslant \left\lfloor \frac{1}{2} \lfloor (d-1)/2 \rfloor \right\rfloor + 1$. Z drugiej strony lemat 3.4 pokazuje, że $\mathrm{MIS}(G[\mathcal{S}]) \leqslant \frac{4(a+1)^2}{\pi}$, więc

$$|U| \leqslant \frac{4(a+1)^2}{\pi} \left(\left\lfloor \frac{1}{2} \lfloor (d-1)/2 \rfloor \right\rfloor + 1 \right)^{-1} \leqslant \frac{16(a+1)^2}{\pi d}.$$

Na zakończenie zauważmy, że części (a) i (b) kroku (2) algorytmu RulingSetInSquare można zaimplementować używając procedury przeszukiwania wgłąb

BreadthFirstSearch, która działa w $O(diam(G[\mathcal{S}]))$ rundach synchronicznych. Z lematu 3.4 wiemy, że $diam(G[\mathcal{S}]) = O(a^2)$, stąd czas działania algorytmu wynosi $O(a^4/d)$ rund synchronicznych. □

Jak wspomniano powyżej, w drugim i trzecim kroku algorytmu RulingSet wyeliminowane są lokalne kolizje, czyli usunięte zostają z R wszystkie wierzchołki o odległości mniejszej bądź równej d. Zrealizujemy to korzystając z następującej procedury.

DeletingLocalCollisions
Wejście: Spójny graf G, $M \subseteq V(G)$ oraz ustalona stała d.
Wyjście: Podzbiór $M' \subseteq M$
(1) Niech $M' := \emptyset$ i niech $Z := M$ oznacza wierzchołki, które mogą być dodane do M'.
(2) Dopóki $Z \neq \emptyset$ wykonaj:
  (a) Wybierz wierzchołek w z $Z \setminus M'$ o najmniejszym identyfikatorze ID i $M' := M' \cup w$, $Z := Z \setminus w$.
  (b) Usuń z Z wszystkie wierzchołki $u \in Z$ takie, że $dist(w, u) \leqslant d$.
(3) Zwróć M'.

Lemat 3.7. *Jeśli M jest zbiorem $(0, b)$–silnym, czyli zbiorem b-dominującym, wtedy M' wygenerowany przez algorytm* DeletingLocalCollisions *jest $(d + 1, d + b)$–silny. Co więcej algorytm* DeletingLocalCollisions *znajduje wynikowy zbiór M' w co najwyżej $(O(|V(R)|a^2))$ synchronicznych rundach.*

Dowód. Dla dowolnych wierzchołków v, w ze zbioru M' mamy $dist(v, w) > d$, gdyż po dodaniu wierzchołka do zbioru M', nie może zostać już dodany do M' żaden wierzchołek będący w odległości co najwyżej d od niego (wszystkie takie wierzchołki są usunięte z Z).

Oznaczmy $dist(v, M') = min_{w \in M'} dist(v, w)$. Dla każdego $v \in M$ istnieje wierzchołek $w \in M'$ taki, że $dist(v, w) \leqslant d$. Jeśli taki v nie istnieje wtedy algorytm DeletingLocalCollisions doda wierzchołek v do M' w części (a) kroku (2). Stąd wynika, że dla każdego $v \in M$ zachodzi $dist(v, M') \leqslant d$. Ostatecznie, ponieważ M jest zbiorem b–dominującym, więc M' jest zbiorem $d + b$–dominującym.

Złożoność obliczeniowa wynika z tego, że w części (a) kroku (2) algorytm DeletingLocalCollisions usuwa jeden wierzchołek z Z, czyli łącznie algorytm wykonuje co najwyżej $|M|$ iteracji. W każdej takiej iteracji, używamy procedury przeszukiwania wszerz, by znaleźć wierzchołek o najmniejszym identyfikatorze ID w podgrafie grafu G. Tak samo w części (b) jest wykorzystywana ta sama

procedura. Lemat 3.4 gwarantuje, że jednokrotne wykonanie punktu (2) zajmuje co najwyżej $O(a^2)$ rund synchronicznych. □

Przypomnijmy, że w algorytmie RulingSet, najpierw znajdujemy zbiór $(d+1, d)$–silny w kwadratach $\mathcal{S}_a^{(0,0)}$. Następnie przesuwamy siatkę dwukrotnie o wektory odpowiednio $(a/3, a/3)$ oraz $(2a/3, 2a/3)$, jak pokazano na rysunku 3.1. Po każdym takim przesunięciu usunięte będą kolizje wewnątrz kwadratów za pomocą DeletingLocalCollisions. Poniżej pokażemy, że po pierwszym przesunięciu w każdym kwadracie otrzymamy zbiór $(d+1, 2d)$–silny, a w wyniku drugiego przesunięcia siatki znajdziemy zbiór $(d+1, 3d)$–silny w całym grafie.

Dowód twierdzenia 3.2. Aby pokazać, że zbiór R'' skonstruowany przez algorytm RulingSet jest rzeczywiście zbiorem $(d+1, 3d)$–silnym, zacznijmy od sprawdzenia, czy R'' spełnia pierwszy warunek definicji 3.1. Przyjmijmy, że istnieją dwa wierzchołki $v_1, v_2 \in R''$ takie, że $dist(v_1, v_2) \leqslant d$. Zatem istnieje ścieżka p_{v_1,v_2} o długości co najwyżej d pomiędzy v_1 a v_2. Potraktujmy p_{v_1,v_2} jako krzywą na płaszczyźnie. Zauważmy, że ścieżka ta musi przecinać $L_a^{(0,0)}$ $L_a^{(a/3,a/3)}$ i $L_a^{(2a/3,2a/3)}$. Gdyby ścieżka p_{v_1,v_2} nie przecinała siatki $L_a^{(0,0)}$, leżałaby wewnątrz jednego z kwadratów $\mathcal{S}_a^{(0,0)}$, co stanowiłoby sprzeczność z faktem, że v_1 i v_2 należą do R. Sprzeczność wynika, to z lematu 3.6, który mówi, że R jest zbiorem $(d+1, d)$–silnym. Korzystając z analogicznego rozumowania i lematu 3.7 można wykazać, że ścieżka p_{v_1,v_2} przecina również $L_a^{(a/3,a/3)}$ i $L_a^{(2a/3,2a/3)}$.

Bez straty ogólności załóżmy, że p_{v_1,v_2} najpierw przecina $L_a^{(a/3,a/3)}$ w punkcie q_1, potem przecina $L_a^{(0,0)}$ w punkcie q_0 a na końcu p_{v_1,v_2} przecina siatkę $L_a^{(2a/3,2a/3)}$ w punkcie q_2. Z lematu 3.3 wynika, że $\|q_1, q_0\| + \|q_2, q_0\| \geqslant a/3$. Ponieważ $\|v, w\| \leqslant dist(v, w)$, tak więc $\|q_1, q_0\| + \|q_2, q_0\| \leqslant dist(v_1, v_2)$ co z kolei implikuje $a/3 \leqslant dist(v_1, v_2) < d+1$. To z kolei stanowi sprzeczność z założeniem, że $a = 3d+3$.

Spełnienie drugiego warunku definicji zbioru $(d+1, 3d)$–silnego wynika bezpośrednio z lematów 3.6 i 3.7. Oznaczmy przez $dist(v, R) = min_{r \in R} dist(v, r)$ i zauważmy, że po kroku (1) algorytmu RulingSet dla każdego $v \in G$ mamy $dist(v, R) \leqslant d$, gdyż z lematu 3.6 zbiór R jest d–dominujący w każdym grafie $G[\mathcal{S}]$, $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$. Analogicznie z lematu 3.7 wynika, że odpowiednio po drugim i trzecim kroku algorytmu RulingSet dla każdego $v \in G$ mamy $dist(v, R') \leqslant 2d$ i $dist(v, R'') \leqslant 3d$.

Z lematu 3.6 krok (1) algorytmu RulingSet zajmuje $O\left(a^4/d\right)$ rund synchronicznych. Wiemy również, że w każdym $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$ wynikowy zbiór M' algorytmu RulingSetInSquare jest wielkości $O(a^2/d)$. Każdy $\mathcal{S} \in \mathcal{S}_a^{(a/3,a/3)}$ zawiera wynikowe zbiory zawarte w czterech kwadratach $\mathcal{S} \in \mathcal{S}_a^{(0,0)}$. Stąd z lematu 3.7 wynika, że kroki (2) i (3) algorytmu RulingSet zajmują $4\,|R|\,(O(a^2) + |d|) =$

$O\left(a^2/d\right)\left(O(a^2)\right)$ rund synchronicznych. Ponieważ założyliśmy, że $a = 3d + 3$, stąd RulingSet ma złożoność czasową $O\left(d^3\right)$. □

3.3. Zastosowania silnego zbioru

W pracy [23] Fernandess i Malkhi zaprezentowali algorytm rozproszony, który znajduje aproksymację najmniejszego zbioru k-dominującego (ang. minimal k dominating set) w jednostkowym grafie dyskowym G ze współczynnikiem aproksymacji $O(k)$ w czasie $O(V(G))$ rund synchronicznych. Algorytm ten działa w modelu obliczeń LOCAL+COORD. W tym podrozdziale korzystając z konstrukcji zbioru silnego poprawiamy czas działania tego algorytmu, utrzymując ten sam współczynnik aproksymacji jak Fernandess i Malkhi.

Twierdzenie 3.8. *Niech G będzie jednostkowym grafem dyskowym. Istnieje deterministyczny algorytm, który aproksymuje najmniejszy k–dominujący zbiór grafu G ze współczynnikiem aproksymacji $O(k)$ w czasie $O(poly(k))$ rund synchronicznych.*

Dowód. Najpierw udowodnimy, że jeśli $d \geqslant 1$ wtedy zbiór $(d+1, 3d)$–silny jest $O(d)$ aproksymacją najmniejszego $3d$–dominującego zbioru.

Bezpośrednio z definicji zbioru $(d+1, 3d)$–silnego wynika, że jest on zbiorem $3d$–dominującym . Niech H będzie optymalnym zbiorem $3d$–dominującym i niech $h \in H$. Oznaczmy przez R wynikowy zbiór $(d+1, 3d)$–silny dany przez algorytm RulingSet.

Niech $N_k(h) = \{v \in V(G) : dist(v, h) \leqslant k\}$. Pokażmy, że zachodzi $|N_{3d}(h) \cup R| = O(d)$, co implikuje $|R| = O(d)\,|H|$. Oznaczmy przez $C_r(v)$ koło o środku w punkcie, który odpowiada wierzchołkowi v i o promieniu r. Zauważmy, że dla dowolnych $g_1, g_2 \in \mathrm{MIS}(N_{3d}(h))$, koła $C_{0.5}(g_i)$ i $C_{0.5}(g_j)$ są rozłączne i leżą w kole o środku w h_i i promieniu $3d+1$. Stąd

$$|\mathrm{MIS}(N_{3d}(h)| \leqslant \frac{\pi(3d+1)^2}{\pi/4} = 4(3d+1)^2.$$

Niech $r \in R$ i oznaczmy $\tilde{N}_{v_i} = N_{\lfloor (d-1)/2 \rfloor}(v_i)$. Jeżeli mamy różne $v_i, v_j \in N_{3d}(h) \cup R$ wtedy, z lematu 3.5 wynika że, zbiory $\tilde{N}_{v_i}$ i $\tilde{N}_{v_j}$ są rozłączne i nie istnieje krawędź pomiędzy nimi. Co więcej $\mathrm{MIS}(\tilde{N}_{v_i}) \geqslant \left\lfloor \frac{1}{2} \lfloor (d-1)/2 \rfloor \right\rfloor + 1$ a zatem mamy

$$|N_{3d}(h) \cup R| \leqslant \frac{|\mathrm{MIS}(N_{3d}(h))|}{min_{v_i \in R} \left|\mathrm{MIS}(\tilde{N}_{v_i})\right|} \leqslant \frac{4(3d+1)^2}{\left\lfloor \frac{1}{2} \lfloor (d-1)/2 \rfloor \right\rfloor + 1} = O(d).$$

Podobnie, można udowodnić, że zbiór $(d+1, 3d)$–silny jest $O(d)$ aproksymacją zarówno zbioru $3d+1$-dominującego jak i zbioru $3d+2$–dominującego.

Ostatnią rzeczą jest pokazanie, że możemy w czasie stałym znaleźć stałą aproksymację zbiorów 1–dominujących i 2 dominujących. W tych przypadkach przeprowadźmy odmienną konstrukcję. Problem znalezienia zbioru 1-dominującego w jednostkowych grafach dyskowych został rozpatrzony przez Czyzowicz, Dobrev, Fevens i innych w pracy [17]. Pokazano tam rozproszony algorytm znajdujący w $O(1)$ rundach synchronicznych w modelu LOCAL+COORD $O(1)$ aproksymację najmniejszego zbioru 1-dominującego. Używając analogicznych argumentów jak w pierwszej części tego dowodu możemy pokazać, że najmniejszy 1-dominujący zbiór w jednostkowych grafach dyskowych jest $O(1)$ aproksymacją najmniejszego 2-dominującego zbioru.

Tak więc dla dowolnego k możemy pokazać rozproszony algorytm znajdujący $O(k)$ aproksymację zbioru k dominującego w czasie $O(poly(k))$ synchronicznych rund. □

W pracy [16] Czygrinow i Hańćkowiak zaprezentowali rozproszone algorytmy znajdujące przybliżone rozwiązania problemów największego skojarzenia i najmniejszego zbioru niezależnego w jednostkowych grafach dyskowych. Algorytmy te są deterministyczne i działają w modelu LOCAL w czasie $O(poly(\log(V(G))))$ rund synchronicznych. Współczynnik aproksymacji tych algorytmów wynosi $1+\epsilon$, gdzie ϵ jest pewną stałą. Poniżej pokażemy, że w modelu LOCAL+COORD obliczeń, możemy rozwiązać dwa powyższe problemy w czasie stałym.

Twierdzenie 3.9. *Niech c będzie ustaloną stałą. Istnieją rozproszone deterministyczne algorytmy znajdujące największe skojarzenie i najmniejszy spójny zbiór dominujący w jednostkowych grafach dyskowych z błędem aproksymacji ϵ. Algorytmy te pracują w czasie $poly(\frac{1}{\epsilon})$ rund synchronicznych.*

Dowód. Dwie podprocedury z pracy [16] trwają więcej niż stała liczba rund synchronicznych. Ich implementacja w czasie stałym prowadzi do rozwiązania problemów największego skojarzenia i najmniejszego zbioru niezależnego w czasie stały.

Pierwsza podprocedura związana jest z budową pomocniczego grafu opartego na znalezieniu maksymalnego zbioru niezależnego. W podprocedurze tej w [16] został użyty algorytm z pracy [45]. Znajduje on maksymalny zbiór niezależny w dowolnym grafie w czasie $O(\log \Delta(G) log^*(V(G)))$ rund synchronicznych. Jednakże korzystając z założenia, że każdy wierzchołek zna swoje współrzędne na płaszczyźnie (czyli pracując w modelu LOCAL+COORD) można zastosować zamiast algorytmu z [45], algorytm podany w pracy [17], który znajduje maksymalny zbiór niezależny w stałym czasie.

Drugą podprocedurą, która w pracy [16], zajmuje $O(poly(\log(V(G))))$ synchronicznych rund, jest podprocedura klastrowania oparta na konstrukcji silnego zbio-

ru. W pracy [16] użyto do tej konstrukcji algorytmu Awerbuch, Goldberg, Luby i Plotkin z pracy [3]. Algorytm ten znajduje zbiór $(d, d\log|V(G)|)$–silny w dowolnym grafie G w czasie $O(d\log|V(G)|)$ rund synchronicznych, gdzie d jest z góry daną stałą. W przypadku gdy ograniczymy się do grafów dyskowych, w których każdy wierzchołek zna swoją współrzędną, możemy wykonać algorytm RULING-SET, który znajduje zbiór $(d+1, 3d)$–silny w grafie G w stałym czasie.

Te dwie główne modyfikacje podprocedur zamieniają algorytmy zawarte w [16] na nową procedurę działającą w czasie $poly(\frac{1}{\epsilon})$. □

3.4. Regularne klastrowanie

Jednym z uogólnień problemu klastrowania grafów jest problem regularnego klastrowania grafów (ang. regular clustering) czyli podziału grafu na takie składowe spójności, których liczba wierzchołków mieści się w pewnym, z góry określonym, przedziale. Dotychczas znane były tylko algorytmy, które dzieliły w sposób rozproszony graf na klastry o małej średnicy (bez ograniczeń na liczbę wierzchołków w klastrze). Wyniki te można znaleźć w pracach [1, 4, 8, 19, 51, 61] i [3].

W niniejszym rozdziale zaprezentowany zostanie rozproszony algorytm, który znajduje regularne klastrowanie jednostkowego grafu dyskowego. Algorytm ten ma na wejściu dowolny graf dyskowy G i z góry dany stały parametr λ, wyjście zaś stanowi podział G na rozłączne spójne podgrafy $H_1, H_2, \ldots H_k$ takie, że dla dowolnego $1 \leqslant i \leqslant k$ zachodzi nierówność $\lambda \leqslant |V(H_i)| \leqslant 5\lambda - 4$. Pokażemy, że prezentowany algorytm znajduje wynik w $O(\lambda^3)$ synchronicznych rundach i udowodnimy, że ograniczenia górnego $5\lambda - 4$ nie można poprawić. Rozszerzymy również przedstawiony rezultat pokazując jak można uzyskać regularne klastrowanie w grafach o ograniczonym stopniu (ang. bounded degree graphs) i w grafach o ograniczonym wzroście (ang. bounded growth graphs).

Jak się okazuje nasza konstrukcja regularnego klastrowania ta ma wiele zastosowań. Załóżmy, że dla pewnej funkcji f znaleziono podział sieci na spójne klastry których wielkość jest co najmniej λ a co najwyżej $f(\lambda)$. Po pierwsze konstrukcja ta znacząco upraszcza protokoły komunikacyjne (ang. routing protocol). Mając dane regularne klastrowanie w każdym klastrze znajdujemy lidera, który zbiera całą informację ze wszystkich wierzchołków swojego klastra po czym komunikacja w całej sieci, odbywała się pomiędzy liderami. Z własności regularnego klastrowania wynika, że pojedynczy lider otrzyma informacje od co najmniej λ wierzchołków a jednocześnie od co najwyżej $f(\lambda)$ wierzchołków. Oznacza to, że duża część zasobów lidera zostanie użyta, a jednocześnie lider będzie obsługiwać co najwyżej $f(\lambda)$ połączeń.

Drugim zastosowaniem jest rozwiązywanie problemów, w których wymaga się, by co najmniej λ wierzchołków pracowało równolegle nad tym samym problemem. Rozważać można model, w których pojedynczy wierzchołek może popełnić błąd w obliczeniach i dopiero stwierdzenie, że większość z λ wierzchołków uzyskała ten sam wynik, daje nam zadowalającą pewność obliczeń. Inną kwestią jest zebranie i analiza statystyczna danych z wierzchołków (np. sensorów zbierającymi informacje o temperaturze). Okazuje się, że lokalne zebranie co najmniej λ danych jest potrzebne do wiarygodnego obliczenia średniej. Z drugiej jednak strony analizowanie więcej niż $f(\lambda)$ próbek może okazać się zbyt złożone obliczeniowo.

Po trzecie zauważmy, że regularne klastrowanie optymalizuje pod względem długości komunikatów i liczby przesyłanych informacji wiele istniejących algorytmów rozproszonych zawierających procedurę klasycznego rozproszonego klastrowania.

Podkreślmy również, że problem ten jest uogólnieniem problemu skojarzenia doskonałego (skojarzenie doskonałe to regularne klastrowanie w którym $\lambda = 2$ i $f(\lambda) = 2$). Tak więc znalezienie regularnego klastrowania z dowolnie przyjętymi parametrami λ oraz $f(\lambda)$ jest zagadnieniem NP-trudnym.

Prezentowany algorytm na jednostkowych grafach dyskowych w rozproszonym, synchronicznym modelu obliczeń LOCAL+COORD. Warto tu zauważyć, że mimo iż zaprezentowany wynik przedstawiony jest w synchronicznym, to istnieje jego prosty odpowiednik w modelu asynchronicznym. Ponadto w twierdzeniu 3.15 pokażemy, jak pogarszając czas działania algorytmu można pominąć założenie o tym, że każdy wierzchołek zna swoje położenie na płaszczyźnie.

Przedstawiony algorytm RegularClustering składa się z dwóch głównych kroków. W pierwszym znajdujemy podział jednostkowego grafu dyskowego na „duże" klastry o małej średnicy. Taki podział nazwiemy zbalansowanym klastrowaniem. W drugim kroku równolegle wykonujemy podział każdego „dużego" klastra na małe regularne klastry. Do skonstruowania podziału grafu na „duże" klastry o małej średnicy użyjemy algorytmu z rozdziału 3.2 oraz algorytmu zawartego w pracy Schneider i Wattenhofer [64]. Pierwszy z tych algorytmów działa w modelu LOCAL+COORD, drugi natomiast w modelu LOCAL. Oba te algorytmy oznaczymy jako RulingSet. Algorytm RulingSet znajduje zbiór odseparowanych od siebie liderów. Bazując na zbiorze tych liderów znajdziemy zbalansowanym klastrowanie grafu. Poprawność drugiego kroku algorytmu RegularClustering, czyli podziału na regularne klastry, będzie wynikać z pewnych specjalnych własności klastrowania wygenerowanego przez RulingSet.

Wejściowe parametry algorytmu RegularClustering to: ustalony parametr λ oraz jednostkowy graf dyskowy G, który zawiera co najmniej λ wierzchołków. Przedstawiony w tym rozdziale algorytm znajduje w $O(\lambda^3)$ synchronicznych run-

dach klastry (czyli krawędziowo indukowane spójne podgrafy) $H_1, H_2, \ldots H_l$ o następujących własnościach:

1. $H_1, H_2, \ldots H_l$ są spójne i parami rozłączne,
2. $\bigcup_{1 \leqslant i \leqslant l} V(H_i) = V(G)$,
3. Dla każdego $1 \leqslant i \leqslant l$, $\lambda \leqslant |V(H_i)| \leqslant 5\lambda - 4$

Co więcej, okazuje się, że liczba klastrów wielkości pomiędzy $4\lambda - 3$ i $5\lambda - 4$ jest bardzo małą frakcją liczby wszystkich składowych (patrz twierdzenie 3.15).

Pokażemy teraz jak mając dany zbór silny liderów skonstruować można zbalansowane klastrowanie. Niech $R = \left\{r_1, r_2, \ldots, r_{|R|}\right\}$ oznacza zbiór (d, b)-silny (patrz definicja 3.1) a $H_1, H_2, \ldots, H_{|R|}$ będzie ciągiem podgrafów G oznaczającym zbalansowane klastrowanie. Przejście z R do $H_1, H_2, \ldots, H_{|R|}$ polega na tym, że $v \in V(H_i)$ wtedy i tylko wtedy gdy r_i jest najbliższy (spośród wierzchołków z R) do v. Jeśli dla danego v istnieje klika najbliższych wierzchołków z R to v wybiera dowolnego z nich.

Lemat 3.10. *Ustalmy stałe d i b. Niech G będzie grafem spójnym , $|V(G)| \geqslant \frac{d-1}{2}$ i niech R będzie (d, b)-silnym zbiorem w grafie G. Wtedy każdy H_i ma co najmniej $\frac{d-1}{2}$ wierzchołków.*

Dowód. Jeśli $|R| = 1$, wtedy $H_1 = G$. Załóżmy więc, że $|R| > 1$. Dla dowolnego $r \in V(G)$ zdefiniujmy $N_t(r) = \{v \in V(G) : dist(v, r) \leqslant t\}$. Naturalnie dla dwóch różnych wierzchołków $r_i, r_j \in R$ zbiory $N_{\frac{d-1}{2}}(r_i)$ i $N_{\frac{d-1}{2}}(r_j)$ są rozłączne. W przeciwnym przypadku istniała by ścieżka długości mniejszej niż d pomiędzy r_i i r_j. Stanowi to sprzeczność z założeniem, że R jest (d, b)-silnym zbiorem.

Ponieważ G jest spójny, więc każda para wierzchołków z R jest połączona ścieżką. Stąd w każdym ze zbiorów $N_{\frac{d-1}{2}}(r_i)$, gdzie $r_i \in R$, istnieje wierzchołek v taki, że $dist(v, r_i) = \left\lfloor \frac{d-1}{2} \right\rfloor$. Zatem średnica podgrafu G indukowanego przez wierzchołki z $N_{\frac{d-1}{2}}(r_i)$ jest co najmniej $\left\lfloor \frac{d-1}{2} \right\rfloor + 1 \geqslant \frac{d-1}{2}$. Korzystając z faktu, że $N_{\frac{d-1}{2}}(r_i) \subseteq V(H_i)$ otrzymujemy, iż każdy klaster H_i ma co najmniej $\frac{d-1}{2}$ wierzchołków. □

Twierdzenie 3.11. *Niech $\alpha > 2$ będzie danym parametrem i niech G będzie jednostkowym grafem dyskowym, w którym wszystkie składowe spójności mają co najmniej α wierzchołków. W modelu obliczeń LOCAL możemy znaleźć podział G na duże klastry $H_1, H_2, \ldots H_l$ z następującymi własnościami:*

1. *Każdy H_i jest spójny,*
2. *Dla każdego $1 \leqslant i \leqslant l$ mamy $\alpha \leqslant |V(H_i)|$,*
3. *Dla każdego $1 \leqslant i \leqslant l$ mamy $diam(H_i) \leqslant 4\alpha + 2$.*

Powyższy podział zajmuje $O(\log \alpha(\alpha^4 + \log^* |V(G)|))$ *rund synchronicznych.*

Uwaga 3.12. *W modelu LOCAL+COORD powyższe twierdzenie można wzmocnić. Czas działania wynosi* $O(\alpha^3)$ *rund synchronicznych a własność* 3 *przyjmuje następującą postać: dla każdego* $1 \leqslant i \leqslant l$ *mamy* $diam(H_i) \leqslant 12\alpha$.

Dowód. Przypomnijmy, że w modelu LOCAL przedstawiliśmy w uwadze 3.17 znajdujący zbiór $(d+1, d+1)$–silny R. Algorytm ten działa dla grafów ograniczonego wzrostu (klasa ta zawiera grafy dyskowe) a jego czas działania wynosi $O(\log d(d^4 + \log^* |V(G)|))$ rund synchronicznych. Niech $d = 2\alpha$, wtedy z lematu 3.10 wynika, że dla każdego H_i zachodzi $\frac{(2\alpha+1)-1}{2} = \alpha \leqslant |V(H_i)|$. Zbiór R jest $2\alpha+1$-dominującym zbiorem, tak więc każdy klaster H_i ma średnicę co najwyżej $4\alpha + 2$.

W modelu obliczeń LOCAL+COORD analogicznie używamy algorytmu RulingSet (zaprezentowanego w rozdziale 3.2), który dla dowolnej stałej d oraz jednostkowego grafu dyskowego G, znajduje $(d+1, 3d)$-silny zbiór R w $O(d^3)$ rundach synchronicznych. □

Niech H będzie klastrem skonstruowanym przez algorytm RulingSet. Zauważmy, że istnieje co najmniej jeden wierzchołek w H taki, że zbiór jego sąsiadów w H może być podzielony na trzy zbiory A^1, A^2, A^3 o własności, że dla wszystkich i, $1 \leqslant i \leqslant 3$, podgraf H indukowany przez wierzchołki z A^i jest kliką. Przykładem wierzchołka z H o tej własności jest wierzchołek o najmniejszej współrzędnej x (patrz rysunek 3.2). Przyporządkujmy więc H dokładnie jeden taki wierzchołek i oznaczmy go przez $\xi(H)$. Zauważmy, że w modelu LOCAL $\xi(H)$ może być znaleziony w $O(diam(H))$ synchronicznych rundach. Początkowo każdy wierzchołek z H lokalnie sprawdza czy warunek podziału jego sąsiedztwa na trzy kliki jest spełniony a następnie $\xi(H)$ jest wybrany spośród wszystkich wierzchołków spełniających ten warunek.

Mając dany $\xi(H)$ zdefiniujmy zbiór

$$dis_j(H) = \{v \in H : dis(v, \xi(H)) = j\}.$$

Niech $j > 0$ oraz $v \in dis_j(H)$ a $N(v)$ oznacza zbiór wszystkich sąsiadów v w grafie H. Oznaczmy przez $prev(v)$ wierzchołek z $N(v) \cap dis_{j-1}(H)$ o najmniejszym ID. Oczywiście taki wierzchołek zawsze istnieje. Analogicznie możemy zdefiniować

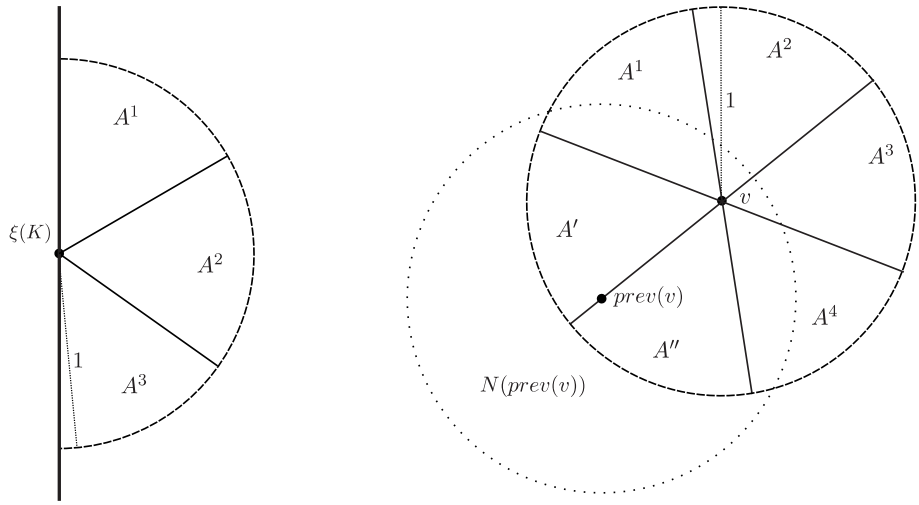
$$next(v) = \{w \in N(v) : prev(w) = v\}.$$

Zauważmy, że mając dane skierowane drzewo BFS skonstruowane przez algorytm BreadthFirstSearch (rozproszona wersja procedury BreadthFirstSearch została zaprezentowana w rozdziale2) grafu H o korzeniu w $\xi(H)$, to dla każdego $v \in H$ wierzchołek $prev(v)$ oraz wierzchołki $next(w)$ mogą zostać wyznaczone w czasie $O(1)$ rund synchronicznych.

Lemat 3.13. *Niech $v \in V(H)$, wtedy zbiór $next(v)$ może być podzielony na co najwyżej cztery zbiory A^1, A^2, A^3, A^4 takie, że każdy z nich indukuje klikę w H.*

Dowód. W przypadku, gdy $v = \xi(H)$ teza twierdzenia bezpośrednio wynika z definicji $\xi(H)$.

Załóżmy że $v \neq \xi(H)$ oraz $v \in dis_j(H)$ dla $j > 0$. Podzielmy koło o środku w v i promieniu 1 na sześć części $A', A'', A^1, A^2, A^3, A^4$ zdefiniowane przez współrzędną wierzchołka $prev(v)$ (w sposób przedstawiony na rysunku 3.2). Dwie z nich A' oraz A'' zawarte są w całości w zbiorze $N(prev(v))$. Zauważmy, że $N(prev(v))$ i $next(v)$ są rozłączne czyli $N(prev(v)) \subseteq dis_j(H) \cup dis_{j-1}(H) \cup dis_{j-2}(H)$ i $next(v) \subseteq dis_{j+1}(H)$, tak więc wszystkie wierzchołki z $next(v)$ leżą w $A^1 \cup A^2 \cup A^3 \cup A^4$. Ponieważ każdy z A^1, A^2, A^3, A^4 ma średnicę geometryczną 1, stąd $next(v)$ może być podzielony na cztery kliki. □



Rysunek 3.2. Dzielenie sąsiedztwa wierzchołka v na kliki.

Niech G będzie grafem i $v \in V(G)$. Oznaczmy przez $G\langle v \rangle$ składową spójności grafu G, która zawiera wierzchołek v.

Przedstawmy teraz algorytm RegularClustering znajdujący regularne klastrowanie w jednostkowym grafie dyskowym.

Twierdzenie 3.14. *Niech $\lambda \geqslant 1$ będzie dowolną stałą i H będzie spójnym jednostkowym grafem dyskowym takim, że $|V(H)| \geqslant \lambda$. Algorytm* RegularClustering *znajduje podział $H'_0, H'_1, \ldots, H'_n$ grafu H taki, że:*

- *$H'_0, H'_1, \ldots, H'_n$ są parami rozłącznymi, spójnymi podgrafami grafu H,*
- *$\bigcup_{0 \leqslant i \leqslant n} V(H'_i) = V(H)$,*
- *$\lambda \leqslant |V(H'_0)| \leqslant 5\lambda - 4$,*
- *Dla wszystkich $1 \leqslant i \leqslant n$, $\lambda \leqslant |V(H'_i)| \leqslant 4\lambda - 3$.*

RegularClustering
Wejście: Stała λ i spójny podgraf H grafu UDG
Wyjście: Graf H' którego składowymi spójności są regularne klastry.
(1) Znajdź $\xi(H)$. Niech $E' = \emptyset$ będzie zbiorem krawędzi i niech H' oznacza graf indukowany przez E' .
(2) Znajdź $dis(v, \xi(H))$ dla wszystkich $v \in H$ używając procedury BreadthFirstSearch.
(3) FOR $j = diam(H) - 1$ DOWN TO 0 DO:
Dla każdego $v \in dis_j(H)$ równolegle wykonaj
(a) dopóki istnieją dwa różne wierzchołki $w, w' \in next(v)$ posiadające cztery poniższe własności:
— $ww' \in E(H)$
— $H' \langle w \rangle \neq H' \langle w' \rangle$
— $|V(H' \langle w \rangle)| < \lambda$
— $|V(H' \langle w' \rangle)| < \lambda$
to ustaw $E' := E' \cup ww'$.
$E_j^{(3a)} := E'$.
(b) jeżeli $w_1, w_2 \dots w_k \in next(v)$ jest zbiorem wierzchołków o dwóch poniższych własnościach:
— dla każdego $1 \leqslant i, j \leqslant k$ i $i \neq j$ zachodzi $H' \langle w_i \rangle \neq H' \langle w_j \rangle$
— dla każdego $1 \leqslant i \leqslant k$ mamy $|V(H' \langle w_i \rangle)| < \lambda$
to podstaw $E' := E' \cup vw_1 \cup vw_2 \cup \dots \cup vw_k$.
$E_j^{(3b)} := E'$.
(4) Jeśli $j = 0$ i $|V(H' \langle \xi(H) \rangle)| < \lambda$ wtedy połącz $H' \langle \xi(H) \rangle$ z dowolną składową $H' \setminus H' \langle \xi(H) \rangle$ używając jednej krawędzi.
(5) Zwróć H' jako wynik.

Algorytm RegularClustering *działa w* $O(diam(H))$ *synchronicznych rundach w modelu LOCAL.*

Dowód. Niech $H_j'^{(3a)}$ będzie grafem indukowanym przez $E_j^{(3a)}$ i niech $H_j'^{(3b)}$ będzie grafem indukowanym przez $E_j^{(3b)}$. Najpierw udowodnijmy, że po kroku $(3a)$ iteracji j-tej każdy $v \in dis_j(H)$ ma co najwyżej czterech sąsiadów $w_1, w_2, w_3, w_4 \in next(v)$ takich, że dla każdego $1 \leqslant i \leqslant 4$ zachodzi:

— Grafy $H_j'^{(3a)} \langle w_i \rangle$ są parami rozłączne.
— $\left|V(H_j'^{(3a)} \langle w_i \rangle)\right| < \lambda$.

Załóżmy nie wprost że jest więcej niż cztery wierzchołki o powyższych własnościach. Zatem korzystając z lematu 3.13 i zasady szufladkowej co najmniej dwa z nich, powiedzmy w_j, w_k, leżałyby w tej samej klice co oznacza, że w_j i w_k połą-

czone byłyby krawędzią w G. Ponieważ $\left|V(H_j'^{(3a)}\langle w_j\rangle)\right| < \lambda$ i $\left|V(H_j'^{(3a)}\langle w_k\rangle)\right| < \lambda$, więc w kroku $(3a)$ algorytmu RegularClustering wierzchołki w_j z w_k zostały połączone krawędzią w H co stanowi sprzeczność.

Pokażmy teraz, że jeśli $0 \leqslant j < diam(H)$ wtedy w j-tym kroku algorytmu, dla każdego v takiego, że $dis(v, \xi(H)) > j$, zachodzi:

— $|V(H_j'^{(3b)}\langle v\rangle)| \leqslant 4\lambda - 3$;
— Jeśli $|V(H_j'^{(3b)}\langle v\rangle)| < \lambda$, wtedy $\exists_{w \in dis_j(H)}$ takie, że $w \in V\left(H_j'^{(3b)}\langle v\rangle\right)$.

Zauważmy, że jeśli w kroku $(3a)$ połączymy krawędzią dwie składowe wielkości mniejszej niż λ, wtedy stworzymy składową wielkości co najwyżej $2\lambda - 1$. Ponieważ w kroku $(3b)$ łączymy co najwyżej cztery składowe wielkości mniejszej niż λ, zatem tak utworzona składowa składa się z co najwyżej $4\lambda - 3$ wierzchołków.

Aby udowodnić drugą własność wystarczy pokazać, że jeśli istnieje $w \in V\left(H_{j+1}'^{(3b)}\right)$ takie, że $\left|V\left(H_{j+1}'^{(3b)}\langle w\rangle\right)\right| < \lambda$, wtedy w j-tym kroku w zostanie z pewnością dodany do składowej z więcej niż λ wierzchołkami w kroku $(3a)$ lub zostanie połączony z $prev(w) \in V\left(H_j'^{(3b)}\right)$.

Ostatnią obserwacją jest to, że w kroku (4) łączymy składową wielkości mniejszej niż λ, ze składową o wielkości pomiędzy λ i $4\lambda - 3$. Tak więc otrzymamy dokładnie jeden klaster H_0' taki, że $\lambda \leqslant |V(H_0')| \leqslant 5\lambda - 4$. Pozostałą część klastrów oznaczmy przez $H_1', H_2', \ldots H_n'$.

Udowodnijmy teraz złożoność czasową. Zauważmy, że kroki $(3a)$ i $(3b)$ algorytmu RegularClustering mogą być zaimplementowane w stałej liczbie kroków synchronicznych. Implementacja ta przebiega następująco:

— Początku wszystkie wierzchołki z $N(v) \cap dis_{j+1}(H)$ wysyłają do v informacje o swoich współrzędnych i wielkości ich składowych (wielkość składowych obliczana jest podczas $j+1$-tej iteracji).
— Następnie v w stałej liczbie rund uruchamia lokalnie krok $(3a)$ i $(3b)$.
— Na zakończenie v wysyła do wierzchołków z $N(v) \cap dis_{j+1}(H)$ informacje o tym które z nich mają się połączyć krawędzią z H'.

Oznacza to, że liczba rund synchronicznych, potrzebnych do wykonania algorytmu jest równa $O(diam(H))$. □

Twierdzenie 3.15. *Niech $c \geqslant 1$ i $\lambda \geqslant 1$ będzie pewną stałą. Niech G będzie spójnym jednostkowym grafem dyskowym takim, że $|V(G)| \geqslant \lambda c$. W modelu LOCAL istnieje rozproszony algorytm znajdujący podział G na klastry o wielkości pomiędzy λ a $5\lambda - 4$. Algorytm ten działa w $O(\log(c\lambda)((c\lambda)^4 + \log^* |V(G)|))$ synchronicznych rundach. W modelu LOCAL+COORD czas jego działania można przyspieszyć do $O((c\lambda)^3)$ rund synchronicznych. Co więcej liczba klastrów o rozmiarze większym niż $4\lambda - 3$ stanowi $O(1/c)$ frakcję liczby wszystkich klastrów.*

Dowód. Zauważmy, że z twierdzenia 3.11 wiemy jak w sposób rozproszony można znaleźć podział jednostkowego grafu dyskowego na klastry $H_1, H_2, \ldots H_l$ o następujących własnościach:

1. Każdy H_i jest spójny,
2. Dla każdego $1 \leqslant i \leqslant l$ mamy $c\lambda \leqslant |V(H_i)|$,
3. Dla każdego $1 \leqslant i \leqslant l$ mamy $diam(H_i) \leqslant 4(c\lambda) + 2$ ($diam(H_i) \leqslant 6c\lambda$ w modelu LOCAL+COORD).

Klastrowanie to może być znalezione w $O(\log(c\lambda)((c\lambda)^4 + \log^* |V(G)|))$ synchronicznych rundach ($O((c\lambda)^3)$ rundach w modelu LOCAL+COORD).

Dla każdego z otrzymanych klastrów uruchamiamy równolegle algorytm RegularClustering. Z twierdzenia 3.14, każdy klaster H_i może być podzielony na regularne klastry, z których co najwyżej jeden ma wielkość pomiędzy $4\lambda-3$ a $5\lambda-4$ natomiast reszta ma wielkość pomiędzy λ and $4\lambda-3$. Ten podział dla konkretnego klastra H_i zajmuje $O(diam(H_i))$ synchronicznych rund. Ponieważ każdy H_i ma co najmniej $c\lambda$ wierzchołków, to liczba klastrów o wielkości pomiędzy $4\lambda-3$ a $5\lambda-4$ jest $O(1/c)$ frakcją wszystkich klastrów. □

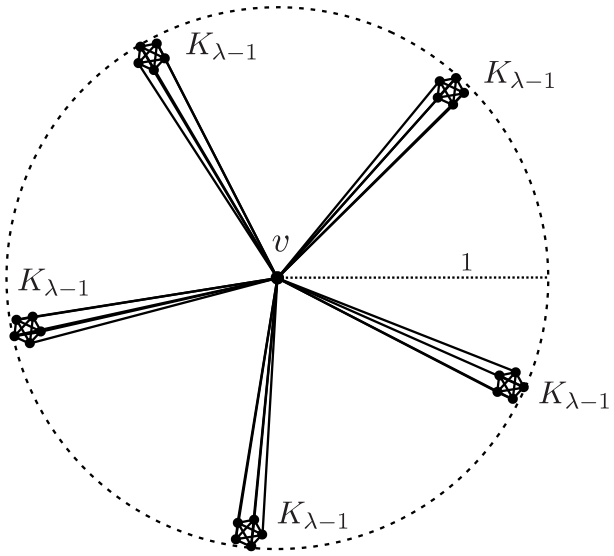
Pokażemy teraz, że ograniczenia $5\lambda - 4$ nie można poprawić. Dokładniej, podamy przykład jednostkowego grafu dyskowego, dla którego nie istnieje podział na klastry $H'_1, H'_2, \ldots H'_l$ takie, że każdy H'_i jest spójny oraz dla każdego $1 \leqslant i \leqslant l$ mamy $\lambda \leqslant |V(H'_i)| < 5\lambda - 4$.

Rozpatrzmy graf z zaprezentowany na rysunku 3.3, zawierający jeden wierzchołek v, połączony z pięcioma klikami $K_{\lambda-1}$. Kliki te znajdują się w odległościach co najmniej 1 od siebie (nie ma pomiędzy nimi krawędzi). Ponieważ każdy klaster jest wielkości większej niż λ, to każda $K_{\lambda-1}$ musi być połączona z v. Otrzymujemy w ten sposób jedną składową o wielkości dokładnie $5(\lambda - 1) + 1$.

Prezentowany algorytm można zmodyfikować tak, żeby konstruował on regularne klastrowanie w dwóch istotnych klasach grafów: grafach ograniczonego stopnia i grafach ograniczonego wzrostu. Najpierw zdefiniujmy potęgę grafu. Pojęcie to potrzebne będzie w dalszych definicjach i dowodach.

Definicja 3.16. *Niech k będzie ustaloną stałą i G będzie grafem. G^k jest k-tą potęgą grafu G jeśli $V(G^k) = V(G)$ oraz krawędzie w G^k łączą te pary wierzchołków, które oddalone są od siebie w G o co najwyżej k.*

Niech Δ będzie maksymalnym stopniem grafu G. Pokażemy teraz jak rozszerzyć algorytm RegularClustering na klasę grafów ograniczonego stopnia, czyli grafów, w których Δ jest ograniczona przez stałą.



Rysunek 3.3. Na tym przykładzie $\lambda = 6$.

Uwaga 3.17. *Niech d będzie stałą i G będzie grafem ograniczonego stopnia. nieznacznie modyfikując algorytm z twierdzenia 3.15 uzyskujemy rozproszony algorytm konstruujący regularne klastrowanie G w $O(\log^* n)$ rundach synchronicznych.*

Dowód. Najpierw zauważmy, że maksymalny zbiór niezależny w G^d , gdzie G^d jest d-tą potęgą grafu G, jest $(d+1, d)$-silnym zbiorem w G. W pracy [31] Goldberg, Plotkin i Shannon wskazali rozproszony algorytm deterministyczny znajdujący maksymalny zbiór niezależny w czasie $O(\log \Delta(\Delta^2 + \log^* n))$ rund synchronicznych. Stąd używając procedurę z [31], możemy znaleźć maksymalny zbiór niezależny w G^d w czasie $O(\log \Delta^d(\Delta^{2d} + \log^* n))$ rund synchronicznych. Tak więc modyfikacja algorytmu RegularClustering, polegająca na dowolnym wybraniu wierzchołka $\xi(H)$ sprawi, że uzyskamy algorytm znajdujący podział G na klastry o wielkościach pomiędzy d a $\Delta(d-1)+1$. Podział ten zajmie $O(d(\Delta^{2d} + \log^* n))=$ $O(\log^* n)$ synchronicznych rund. □

Wskażmy poniżej jak rozszerzyć przedstawiony wynik do klasy grafów o ograniczonym wzroście.

Definicja 3.18. *Niech $N^r(v)$ będzie zbiorem wszystkich wierzchołków odległych o co najmniej r kroków od v. Graf $G = (V, E)$ jest ograniczonego wzrostu, jeśli istnieje wielomianowa funkcja $f(r)$ taka, że dla każdego wierzchołka $v \in V$, wielkość największego zbioru niezależnego w grafie indukowanym przez $N^r(v)$ jest co najwyżej $f(r)$, $\forall_{r \geqslant 0}$. Funkcję f nazywamy funkcją ograniczającą wzrost.*

Klasa grafów ograniczonego wzrostu obejmuje wiele klas grafów. Są nimi np.: jednostkowe grafy dyskowe, quasi-jednostkowe grafy dyskowe (ang. quasi unit disk

graphs) i jednostkowe grafy kulowe (ang. unit ball graphs). W pracy [64] Schneider i Wattenhofer zaprezentowali rozproszony algorytm deterministyczny znajdujący maksymalny zbiór niezależny w grafach ograniczonego stopnia. Czas działania tego algorytmu wynosi $O(\log^* n)$-rund synchronicznych. Rezultat z [64] połączony z konstrukcją zawarta w uwadze 3.17 implikuje fakt.

Fakt 3.19. *Niech G będzie grafem ograniczonego wzrostu z daną funkcją ograniczającą wzrost f. Możemy skonstruować podział grafu G na spójne klastry o wielkościach pomiędzy d a $f(1)(d-1)+1$ w $O(\log(d)(f(d)^2+\log^* n))$ rundach synchronicznych.*

Dowód. Niech G będzie grafem ograniczonego wzrostu z funkcją ograniczającą f. Niech $I = \{v_1, \ldots, v_l\}$ będzie największym zbiorem niezależnym w G. Zauważmy, że największy zbiór niezależny w grafie $G^d[I]$ (czyli podgrafie G^d indukowanym przez I) jest $(d+1, d+1)$-silnym zbiorem. Stąd możemy znaleźć $(d+1, d+1)$-silny zbiór w grafach ograniczonego wzrostu w $O(\log f(d)(f(d)^2+\log^* n))$-synchronicznych rundach. Ponieważ $f(d)$ jest wielomianowe, czas potrzebny do tej konstrukcji jest $O(\log(d)(poly(d)+\log^* n))$. □

4. Algorytmy kolorujące

4.1. Wprowadzenie

Jednym z podstawowych zagadnień dotyczących sieci radiowych jest przydział częstotliwości. Jak już wspomniane zostało w rozdziale 2.3, jeśli dwa elementy sieci nadają równocześnie na tej samej częstotliwości do trzeciego elementu, to może nastąpić interferencja i zakłócenie sygnału. Jednym z rozwiązań tego problemu jest dobranie częstotliwości tak, aby dowolne dwa elementy sieci nadające informację do jednego elementu miały przydzielone różne częstotliwości. Naturalnym pytaniem jest jak dobrać kanały tak, aby zużyć najmniejszą liczbę częstotliwości i uniknąć zakłóceń. Jest to tak zwany problem optymalnego przydziału częstotliwości. W tym rozdziale zajmiemy się problemem z nim powiązanym, a mianowicie zagadnieniem optymalnego kolorowania grafu reprezentującego sieć. Zauważmy, że jeśli różnym kolorom przypiszemy różne częstotliwości, to kolorowanie optymalne wyznacza przydział kanałów, w którym łączące się wierzchołki mają różne częstotliwości.

Rozdział 4.2 poświęcony jest algorytmom rozproszonym kolorującym grafy odpowiadające sieciom radiowym. W szczególności, zaprezentowane zostanie kilka algorytmów rozproszonych, które znajdują stałą aproksymację optymalnego kolorowania jednostkowych grafów dyskowych w stałej liczbie synchronicznych rund. Przeanalizujemy je pod względem współczynnika aproksymacji i zakładanej mocy obliczeniowej.

Rozdział 4.3 poświęcony zostanie modelowi sieci radiowej w której nadajniki mogą zmniejszać swoją moc. Pokazany zostanie rozproszony algorytm, który przydzieli moce do nadajników w taki sposób, że sieć używa minimalnej (z dokładnością do stałej) liczby kolorów, pozostając jednocześnie spójną. Algorytm ten działać będzie w stałej liczbie synchronicznych rund.

Część z prezentowanych rezultatów zawarta jest w pracy [40].

4.2. Kolorowanie grafów dyskowych w stałym czasie

Zagadnienie kolorowania sprowadza się do problemu przyporządkowania wierzchołkom liczb naturalnych (częstotliwości radiowych) w ten sposób, że na przyległych wierzchołkach liczby są różne. Natomiast optymalne kolorowanie to takie, w którym liczba użytych kolorów (liczb) jest najmniejsza. W rozdziale tym przedstawione zostanie kilka bardzo szybkich algorytmów rozproszonych, które znajdują stałe aproksymacje optymalnego kolorowania. Wszystkie prezentowane algorytmy działają w naturalnym modelu sieci z komunikacją radiową – na jednostkowych grafach dyskowych (patrz definicja 1.2) w rozproszonym modelu obliczeń LOCAL+COORD (patrz definicja 1.3).

Analizowane algorytmy różnią się nie tylko pod względem współczynników aproksymacji, ale także zakładanej mocy obliczeniowej wierzchołków. Drugi z wymienionych parametrów określa, jakiego rzędu złożoności obliczeniowej problemy rozwiązuje pojedynczy wierzchołek w jednej rundzie synchronicznej. Dla przykładu, jeśli w jednej rundzie synchronicznej każdy wierzchołek musi zebrać unikalne identyfikatory (ID) wszystkich swoich sąsiadów i wyznaczyć z nich maksimum, to zakładana moc obliczeniowa wynosi $O(\Delta(G))$.

W rozdziale tym zaprezentujemy cztery algorytmy rozproszone: 7heksagons, Greedy7heksagons, 4Stripes, 3heksagons znajdujące w stałej liczbie synchronicznych rund kolorowanie jednostkowych grafów dyskowych. Algorytmy te porównamy z trzema znanymi rozproszonymi algorytmami, które będziemy nazywać DischargingMIS ,Lexicographic i SmallestLast.

Przedstawmy teraz porównania parametrów wszystkich siedmiu wymienionych powyżej algorytmów. W poniższej tabeli, jak również w dalszej części rozdziału $\omega(G)$ oznaczać będzie wielkość największej kliki w grafie G, a $\chi(G)$ – liczbę kolorów w optymalnym kolorowaniu grafu G.

Nazwa algorytmu	Osz. górne	Osz. dolne	Czas dzi.	Moc obl.	Wsp.	Per.
Lexicographic	$3\chi(G)-2$	$2\omega(G)$	$O(V(G))$	$O(\omega(G))$	+	-
SmallestLast	$3\chi(G)-2$	$2\omega(G)$	$O(V(G))$	$O(\omega(G))$	-	-
DischargingMIS	$5\chi(G)-4$	$\frac{10}{3}\omega(G)$	$O(\omega(G))$	$O(\omega(G))$	+	-
7heksagons	$7\chi(G)$	$7\omega(G)$	$O(1)$	$O(\omega(G))$	+	+
Greedy7heksagons	$5\chi(G)-4$	$\omega(G)$	$O(1)$	$O(\omega(G)^2)$	+	-
4Stripes	$4\chi(G)$	$4\omega(G)$	$O(1)$	$O(\omega(G)^3)$	+	+
3heksagons	$3\chi(G)$	$3\omega(G)$	$O(1)$	$O(2^{\omega(G)})$	+	+

Osz. górne - Jest to oszacowanie górne na liczbę kolorów użytych przez dany algorytm.
Osz. dolne - Jest to oszacowanie dolne na liczbę kolorów użytych przez dany algorytm.

Czas dzi. - Określa czas działania, czyli ilość rund synchronicznych potrzebnych do znalezienia wyniku.
Moc obl. - Jest to zakładana moc obliczeniowa w algorytmie.
Wsp. - Mówi czy algorytm wymaga znajomości współrzędnych.
Per. - Określa czy można zoptymalizować rozwiązanie przy pomocy kolorowania permutacyjnego.

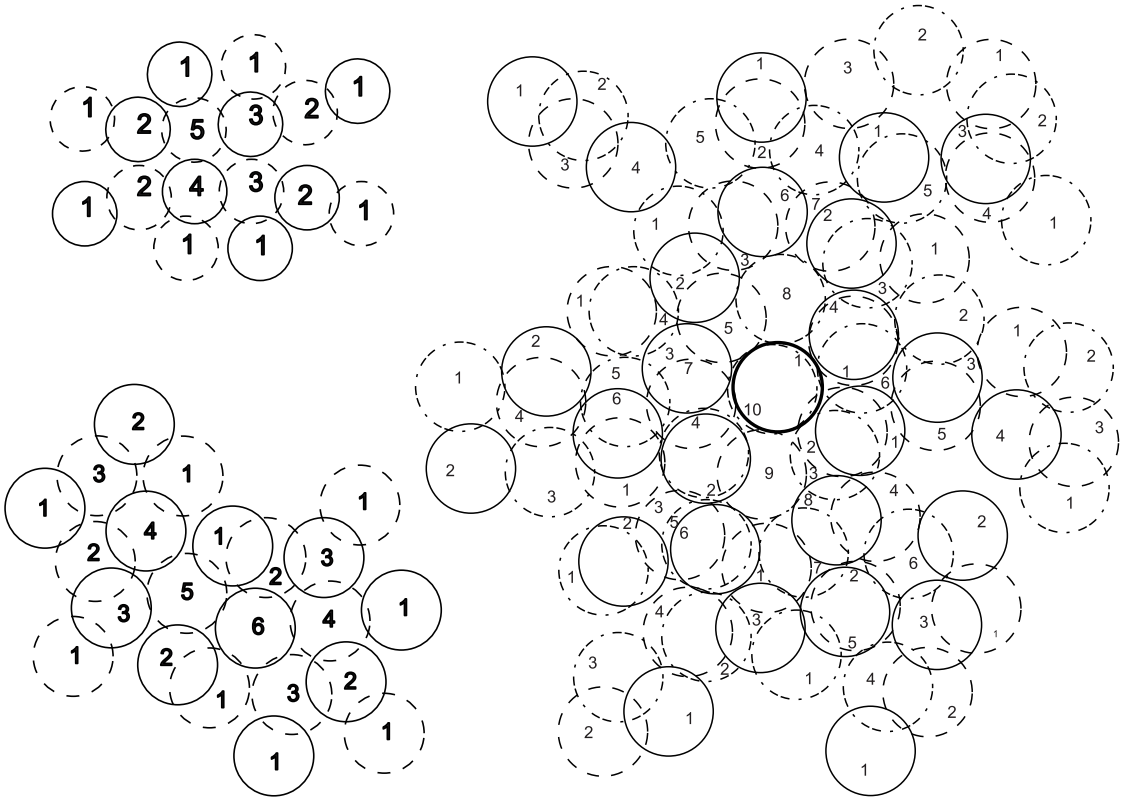
Większość dotychczasowych badań dotyczących kolorowania grafów dyskowych skupiała się na problemach konstrukcji algorytmów i złożoności obliczeniowej w klasycznym modelu obliczeń. W pracy [32] Gräf, Stumpf i Weißenfels pokazali, że problem optymalnego kolorowania jednostkowych grafów dyskowych jest problemem NP-trudnym. Pokazali również, że ograniczeniem dolnym na liczbę chromatyczną, czyli minimalną liczbę kolorów potrzebną do poprawnego kolorowania jest $\omega(G)$. Z drugiej strony najlepszym znanym ograniczeniem górnym na liczbę chromatyczną w UDG jest $3\omega(G) - 2$, zaprezentowane w pracy [32]. Pokazane w niej zostały dwa algorytmy kolorujące ze współczynnikami aproksymacji 3. Algorytmów tych jednak nie można zmodyfikować tak, aby działały w krótkim czasie w modelu rozproszonym.

Przykładem algorytmu działającego w modelu rozproszonym jest algorytm deterministyczny zaprezentowany przez Parthasarathy i Gandhi w pracy [57]. Znajduje on kolorowanie UDG $5\chi(G) - 4$ kolorami w $O(\Delta \log^2 n)$ rundach synchronicznych. Zastosowana w nim metoda polega na znajdowaniu kolejnych maksymalnych zbiorów niezależnych w grafie i nadawanie im kolejnych kolorów. Z kolei w pracy [64] J. Schneider i R. Wattenhofer zaprezentowali rozproszony algorytm deterministyczny znajdujący maksymalny zbiór niezależny w grafach ograniczonego stopnia. Czas działania tego algorytmu wynosi $O(\log^* n)$ rund synchronicznych. Korzystając z tego wyniku i stosując metodę analogiczną do zastosowanej w pracy [57] można uzyskać kolorowanie UDG w $O(\Delta \log^* n)$ synchronicznych rundach w przy użyciu $5\chi(G) - 4$ kolorów.

W modelu rozproszonym, w którym wierzchołki znają współrzędne, znalezienie maksymalnego zbioru niezależnego jest możliwe w stałej liczbie synchronicznych rund dzięki wykorzystaniu algorytmu z pracy [17]. Z pracy tej wynika, że w modelu LOCAL+COORD można pokolorować UDG $5\chi(G) - 4$ kolorami w czasie $O(\Delta)$. W dalszych rozważaniach algorytm ten nazywać będziemy DischargingMIS. Z kolei w pracy [58] Peeters udowodnił, że zachłanne kolorowanie w kolejności leksykograficznej, czyli od wierzchołka o najmniejszej współrzędnej x do wierzchołka o największej współrzędnej x, daje w modelu LOCAL+COORD poprawne kolorowanie przy pomocy $3\omega - 2$ kolorów. Algorytm prezentowany w pracy [58] nazywać będziemy Lexicographic.

Ostatni z algorytmów, który będziemy porównywać do zaprezentowanych w tej rozprawie nazwiemy SmallestLast. Został on wprowadzony w pracy [15], w któ-

rej Couture, Barbeau, Bose, Carmi i Kranakis udowodnili, że istnieje rozproszony algorytm, który w $O(n)$ rundach synchronicznych znajduje kolorowanie UDG przy pomocy $3\omega(G)-2$ kolorów. Algorytm ten ściśle bazuje na pracy [58], ale nie wymaga by wierzchołki znały swoje współrzędne na płaszczyźnie. Z pracy [15] pochodzi również oszacowanie z którego wynika, że zachłanny algorytm kolorujący używa co najmniej $\frac{10}{3}\omega(G)$ kolorów (patrz rysunek 4.1) i co najwyżej 6ω kolorów.

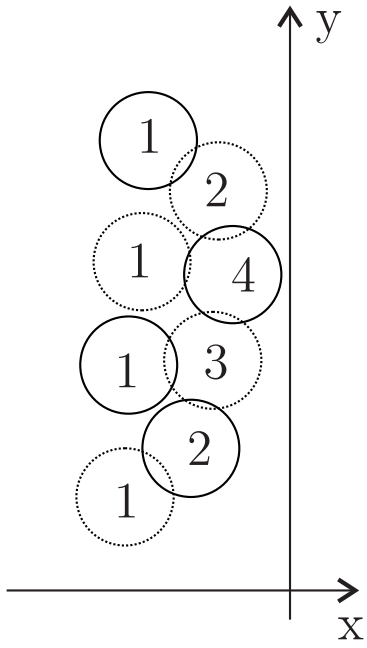


Rysunek 4.1. Oszacowanie na $\frac{5}{2}$, $\frac{6}{2}$ oraz $\frac{10}{3}$ aproksymację optymalnej liczby kolorowania, danego przez algorytm zachłanny.

Nie przedstawiono dotychczas wyników dotyczących oszacowań dolnych na współczynniki aproksymacji algorytmów Lexicographic i SmallestLast. Lukę tę wypełnia nasz lemat:

Lemat 4.1. *Dla przykładu (wskazanego na rysunku 4.2) algorytmy* Lexicographic *oraz* SmallestLast *dają kolorowanie* $2\omega(G)$ *kolorami.*

W przedstawionych pracach nie ma jednak wyników przedstawiających algorytmy kolorujące, które działają w stałej liczbie synchronicznych rund. Tymczasem istnieje naturalna potrzeba stworzenia i analizy takich algorytmów ze względu na



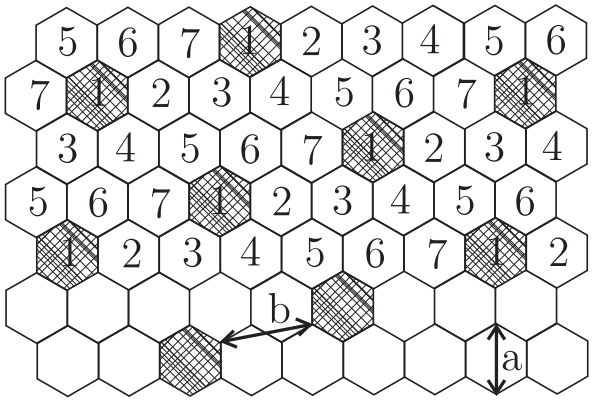
Rysunek 4.2. Przykład dla którego algorytm LEXICOGRAPHIC daje kolorowanie $2\omega(G)$ kolorami.

zastosowania w rozległych sieci radiowych w których zachodzą dynamiczne zmiany struktury. Zaprezentujemy cztery algorytmy rozproszone które w modelu LOCAL+COORD w stałej liczbie synchronicznych rund znajdują stałą aproksymację kolorowania UDG. Według naszej wiedzy są to pierwsze algorytmy kolorujące UDG w tak szybkim czasie.

Pierwszy z prezentowanych algorytmów nosi nazwę 7HEKSAGONS.

7HEKSAGONS
Wejście: Jednostkowy graf dyskowy G.
Wyjście: Poprawne kolorowanie G

1. Podzielmy płaszczyznę na 7 klas heksagonów zgodnie z rysunkiem 4.3 (wartość a wynosi 1).
2. Dla każdego wierzchołka v wykonaj równocześnie:
   a) Niech i oznacza numer klasy heksagonu do której należy v.
   b) Oznaczmy przez $P_i(v)$ wierzchołki będące sąsiadami wierzchołka v znajdujące się w tym samym heksagonie co v.
   c) Wierzchołek v otrzymuje wszystkie identyfikatory ID wierzchołków $P_i(v)$.
   d) v określa liczbę wierzchołków z $P_i(v)$ które posiadają mniejsze ID niż v. Oznaczmy tę liczbę przez x_v.
   e) v otrzymuje kolor $7x_v + i$.



Rysunek 4.3. Podział na 7 klas heksagonów. Na przykładzie $a = 1$ $b > 1$

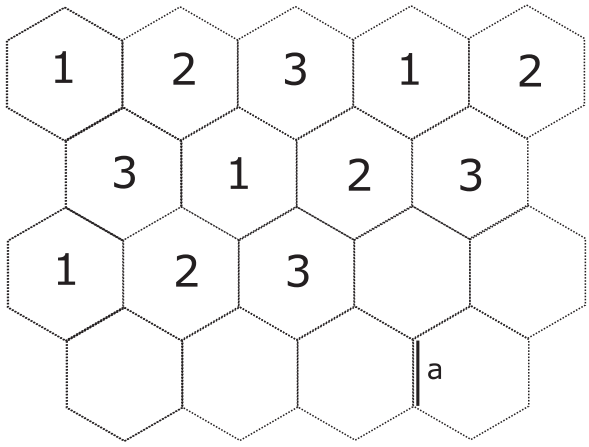
Twierdzenie 4.2. *Algorytm* 7HEKSAGONS *koloruje jednostkowy graf dyskowy G nie więcej niż $7\omega(G)$ kolorami w stałej liczbie rund synchronicznych. Zakładana moc obliczeniowa w jednej rundzie to $O(\omega(G))$.*

Dowód. Zauważmy, że pierwszy krok algorytmu 7HEKSAGONS może być zrealizowany w jednym kroku synchronicznym, gdyż każdy wierzchołek zna swoje współrzędne. W zbiorze $P_i(v)$ wszystkie wierzchołki pokolorują się różnymi kolorami. Wynika to z definicji liczby x_v oraz tego, iż $7x_v + i$ jest funkcją różnowartościową w zbiorze wierzchołków z danego heksagonu. Jeśli $i \neq j$ to v ma kolor $7x_v + i$ a w ma kolor $7x_w + j$. Stąd ich kolory są różne modulo 7. Tak więc jeśli $i \neq j$ to nie będzie także konfliktów kolorów pomiędzy $P_i(v)$ oraz $P_j(w)$. Zauważmy ponadto, że każdy $P_i(v)$ tworzy klikę. Tak więc maksymalny kolor użyty w $P_i(v)$ jest $k = 7\,|P_i(v)| + i \leqslant 7\,|P_i(v)| + 7 \leqslant 7\omega(G)$. Ponieważ w algorytmie każdy wierzchołek musi poznać swoje sąsiedztwo i wyznaczyć w liczbę x_v więc zakładana moc obliczeniowa wynosi $\Delta(G) \leqslant 6\omega(G)$. □

Algorytmem analogicznym do 7HEKSAGONS jest algorytm 3HEKSAGONS. Różnica pomiędzy nimi polega na tym, że dzielimy płaszczyznę tylko na trzy klasy heksagonów o średnicy 2 (patrz rysunek 4.4). Grafy indukowane przez heksagony o średnicy 2 nie stanowią klik, przez co optymalne kolorowanie wewnątrz nich wymaga mocy obliczeniowej rzędu $O(2^{\omega(G)})$.

Uwaga 4.3. *Algorytm* 3HEKSAGONS *koloruje UDG przy użyciu co najwyżej $3\omega(G)$ kolorów. Algorytm ten działa w stałej liczbie synchronicznych rund a zakładana moc obliczeniowa w jednej rundzie to $O(2^{\omega(G)})$.*

Stały czas działania tej procedury wynika z faktu iż średnica UDG wewnątrz heksagonu o boku 1 jest stała. Dowód tego faktu przebiega analogicznie jak dowód lematu 2.7.



Rysunek 4.4. Podział płaszczyzny na 3 klasy heksagonów. Na rysunku $a = 1$.

3HEKSAGONS
Wejście: Jednostkowy graf dyskowy G.
Wyjście: Poprawne kolorowanie G

1. Podzielmy płaszczyznę na 3 klasy heksagonów jak na rysunku 4.4.
2. Zdefiniujmy palety kolorów $P_1 = 1, 4, 7, \ldots$, $P_2 = 2, 5, 8, \ldots$, $P_3 = 3, 6, 9, \ldots$.
3. Każda składowa spójności w każdym heksagonie wybiera sobie lidera.
4. Każdy lider zbiera informacje o swojej składowej (ograniczonej do heksagonu) rozproszonym algorytmem przeszukiwania wszerz BREADTHFIRSTSEARCH.
5. Każdy lider w heksagonie i–tej klasy korzystając z zebranych danych znajduje optymalne kolorowanie swojej składowej przy pomocy palety P_i. Wynikowe kolorowanie rozsyła rozproszonym algorytmem przeszukiwania wszerz BREADTHFIRSTSEARCH. (Rozproszona wersja procedury BREADTHFIRSTSEARCH została zaprezentowana w rozdziale 2.2).

Istnieje możliwość zmniejszenia liczby kolorów użytych w algorytmie 7HEKSAGONS dzięki zastosowaniu procedury kolorowania permutacyjnego (ang. permutative coloring). Kolorowanie permutacyjne zostało opisane w pracy [32]. Najpierw należy pokolorować wierzchołki z wszystkich heksagonów tą sama paletą, to znaczy zamiast funkcji $7x_w + j$ należy wziąć funkcję x_w, a następnie poprawić kolorowanie dla każdej pary klas heksagonów przy pomocy kolorowania permutacyjnego.

Kolorowanie permutacyjne jako wejście bierze dwa rozłączne podzbiory wierzchołków V i W grafu G wraz z ich kolorowaniem. Znajdując maksymalne skojarzenie w grafie dwudzielnym (V, W) permutuje kolory tak aby nie było konfliktów. Dobrą aproksymacje takiego maksymalnego skojarzenia w stałej liczbie synchronicznych rund można uzyskać bazując na twierdzeniu 3.9 z rozdziału 3.2.

Badania eksperymentalne zawarte w [32] pokazują, że dla wielu sieci UDG procedura kolorowania permutacyjnego znacząco poprawia współczynnika aproksymacji kolorowania.

Przeanalizujemy teraz algorytm Greedy7heksagons, który poprawia współczynnik aproksymacji algorytmu 7heksagons. Potrzebuje on większej zakładanej mocy obliczeniowej wierzchołków.

Greedy7heksagons
Wejście: Jednostkowy graf dyskowy G.
Wyjście: Poprawne kolorowanie G

1. Podziel płaszczyznę na 7 klas heksagonów zgodnie z rysunkiem (4.3), gdzie długość a wynosi 1.
2. FOR i:=1 TO 7 DO:
   a) Wybierz w każdym heksagonie i-tej klasy lidera, czyli wierzchołek o najmniejszym ID.
   b) Każdy lider przy pomocy procedury BreadthFirstSearch pobiera informacje o wierzchołkach w swoim heksagonie i kolorach ich sąsiadów (wierzchołki te mogą być już poza heksagonem).
   c) Lokalnie, każdy lider koloruje zachłannie wszystkie wierzchołki w swoim heksagonie uwzględniając wszystkie kolory ich sąsiadów.
   d) Każdy lider rozsyła informacje o przydzielonych kolorach przy pomocy procedury BreadthFirstSearch.

Twierdzenie 4.4. *Algorytm* Greedy7heksagons *koloruje UDG przy pomocy co najwyżej* $6\omega(G) - 5$ *(*$5\chi(G) - 4$*) kolorów w stałej liczbie synchronicznych rund. Zakładana moc obliczeniowa w jednej rundzie to* $O(\omega(G)^2)$

Dowód. Ponieważ heksagony z jednej klasy są oddalone od siebie o co najmniej 1, więc algorytm Greedy7heksagons jest dobrze zdefiniowanym algorytmem zachłannym. W pracy [32] udowodniono, że każdy zachłanny algorytm koloruje UDG przy pomocy co najwyżej $6\omega - 5$ kolorów. Dowód ten korzysta z obserwacji, że sąsiedztwo każdego wierzchołka podzielić można na sześć klik. Co więcej Erlebach i Fiala w pracy [20] pokazali, że algorytm zachłanny na UDG używa $5\chi(G) - 4$ kolorów. Ich dowód bazuje na obserwacji, że wielkość maksymalnego zbioru niezależnego w sąsiedztwie wierzchołka UDG jest co najwyżej 5.

Zakładana moc obliczeniowa wynika z następującego toku rozumowań:

— Wpierw każdy heksagon wybiera swojego lidera. Wymaga to mocy obliczeniowej rzędu $O(\omega(G))$.

— 2-sąsiedztwo badane przez lidera w kroku 2b zawiera co najwyżej $O(\omega(G))$ wierzchołków. Tak więc zebranie informacji o wierzchołkach w 2-sąsiedztwie lidera i wyznaczenie listy wolnych kolorów dla każdego wierzchołka wymaga od lidera mocy obliczeniowej rzędu $O(\omega(G)^2)$.
— Wyznaczenie kolorów dla wszystkich wierzchołków w heksagonie wymaga również od lidera $O(\omega(G)^2)$ operacji.

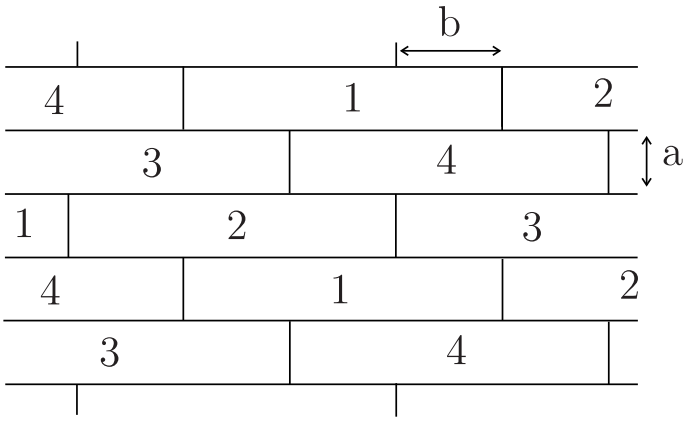
□

W ostatnim z prezentowanych algorytmów kolorowania nazwanym 4Stripes wykorzystamy podział jednostkowego grafu dyskowego na prostokąty szerokości równej $\sqrt{3}/2$. Jak się okazuje graf indukowany na wierzchołkach znajdujących się w jednym prostokącie podziału, może być pokolorowany w sposób optymalny w czasie wielomianowym.

Twierdzenie 4.5. *Podgraf H jednostkowego grafu dyskowego indukowany przez wierzchołki znajdujące się w prostokącie o szerokości co najwyżej $\sqrt{3}/2$ może zostać pokolorowany optymalnie w czasie $|V(H)|\omega(H)^2$.*

Dowód tego twierdzenia znaleźć można w pracy [32] (twierdzenie 4.6). Bazuje on na obserwacji, że H jest grafem koporównywalnym i doskonałym, oraz na konstrukcji maksymalnych przepływów w sieciach.

4Stripes
Wejście: Jednostkowy graf dyskowy G.
Wyjście: Poprawne kolorowanie G
1. Rozpatrzmy podział na 4 klasy prostokątów tak jak na rysunku 4.5.
2. Zdefiniujmy palety kolorów $P_1 = 1, 5, 9, \ldots$, $P_2 = 2, 6, 10, \ldots$, $P_3 = 3, 7, 11, \ldots$, $P_4 = 4, 8, 12, \ldots$.
3. Każda składowa spójności w każdym prostokącie wybiera lidera.
4. Każdy lider zbiera informacje o swojej składowej (ograniczonej do prostokąta) rozproszonym algorytmem przeszukiwania wszerz BreadthFirstSearch.
5. Każdy lider w i-tym prostokącie, korzystając z twierdzenia 4.5, znajduje optymalne kolorowanie swojej składowej przy pomocy palety P_i. Wynikowe kolorowanie rozsyła rozproszonym algorytmem przeszukiwania wszerz BreadthFirstSearch.



Rysunek 4.5. Długość $a = \frac{\sqrt{3}}{2}$ natomiast $b = 1$

Twierdzenie 4.6. *Algorytm* 4STRIPES *koloruje graf UDG przy użyciu co najwyżej* $4\chi(G)$ *kolorów w stałym czasie. Zakładana moc obliczeniowa w jednej rundzie to* $O(\omega(G)^3)$.

Dowód. Dowód poprawności kolorowania wynika z obserwacji, że dwa prostokąty znajdujące się w tej samej klasie, są odległe od siebie o więcej niż 1. Ponieważ klas prostokątów jest 4 a każdy z nich zostaje pokolorowany w sposób optymalny więc użytych kolorów jest co najwyżej $4\chi(G)$.

Wielkość zakładanej mocy obliczeniowej wynika z faktu, że liczba wierzchołków w każdym ograniczonym prostokącie jest $O(\omega(G))$. □

4.3. Kolorowanie w modelu z kontrolą mocy nadajników

W poprzednim rozdziale wspomnieliśmy, że optymalne kolorowanie, jak również optymalny przydział częstotliwości, w jednostkowych grafach dyskowych używa co najmniej $\omega(G) \geqslant \frac{\Delta(G)}{6}$ kolorów, gdzie $\omega(G)$ oznacza wielkość największej kilki, a $\Delta(G)$ maksymalny stopień w grafie G. Co więcej z pracy [32] wynika, że liczba chromatyczna jednostkowego grafu dyskowego G, oznaczana przez $\chi(G)$, jest z góry ograniczona przez $3\omega(G)$. Dla gęstych grafów wartość $\omega(G)$ może okazać się bardzo duża. Z tego powodu nawet optymalne kolorowanie może użyć bardzo dużo kolorów. Dlatego interesującym zagadnieniem jest taka zmiana charakterystyki sieci, by stała się ona rzadsza, a przez to łatwiejsza do pokolorowania. Jednym z podejść do tego zagadnienia jest zmiana mocy nadajników w taki sposób, by promień zasięgu wierzchołków zmniejszył się. Idea takiej zmiany topologii sieci była rozpatrywana w kategoriach redukcji całkowitego zużycia energii w pracach [7],[39],[33],[47],[62],[65],[66],[67]. Zmniejszanie mocy nadajników roz-

patrywano również w kategoriach zmniejszania liczby interferencji (patrz prace [6],[24],[55]). Oczywiście zmniejszanie mocy nadajników optymalizuje zużycie energii w elementach sieci, a przez to wydłuża czas pracy tych elementów. Co więcej, procedura ta redukuje liczbę krawędzi w wynikowym grafie dyskowym, przez co liczba chromatyczna grafu zmniejsza się. Jednakże modyfikacja mocy nadajników może wpłynąć negatywnie na spójność sieci. Po redukcji liczby połączeń, graf może zostać podzielony i niektóre wiadomości mogą nie zostać dostarczone do adresatów. Drugim negatywnym skutkiem zmniejszenia promienia zasięgu nadajników może być znaczące wydłużenie ścieżek, przez które muszą przejść komunikaty, by dotrzeć do swoich adresatów.

W niniejszym rozdziale rozpatrywać będziemy problem kolorowania wierzchołków, oraz przydzielania częstotliwości, w modelu sieci radiowej z dodatkowym założeniem dotyczącym zmiennej mocy nadajników. Pokażemy algorytm działający w rozproszonym modelu obliczeń na jednostkowych grafach dyskowych, który przydziela nadajnikom odpowiednie moce oraz częstotliwości radiowe w ten sposób, by dwa wierzchołki nie nadawały do jednego wierzchołka na tej samej częstotliwości. Zakładać będziemy, że początkowo wszystkie koła odpowiadające nadajnikom mają ten sam promień. Algorytm DiminishPower, zmieniając moc nadajnika, zmniejszać będzie obszar, w którym sygnał nadajnika jest słyszalny, przez co koło odpowiadające zasięgowi nadawania zmniejsza swój promień. Przypomnijmy, że zmniejszenie promieni pociąga za sobą zmniejszenie liczby częstotliwości niezbędnych do przydzielenia. Wynika to z faktu, że sieć staje się rzadsza. Z drugiej strony nadmierne zmniejszenie mocy niektórych nadajników może doprowadzić do sytuacji, gdy pierwotnie spójna sieć rozpadnie się na części, które nie będą mogły się ze sobą komunikować.

Nasz algorytm DiminishPower zmniejszy mocy nadajników bez utraty silnej spójności wynikowego grafu, przy jednoczesnym zmniejszeniu liczby użytych częstotliwości. Pokażemy ponadto, że liczba użytych częstotliwości w przedstawionym algorytmie jest najlepsza z dokładnością do stałej multiplikatywnej.

Według posiadanej przez nas wiedzy prezentowany algorytm jest pierwszym, w którym koncepcja zmiany mocy nadajników jest rozpatrywana w sposób rozproszony i w terminach przydziału częstotliwości. Co więcej, w analizie algorytmu skupiamy się nie tylko na liczbie użytych kolorów, lecz także na całkowitej energii, która została zaoszczędzona w wyniku przedstawionej procedury.

Rozpocznijmy od zdefiniowania problemu przydziału częstotliwości. Kolorowanie grafu nazywać będziemy *przydziałem częstotliwości* jeżeli dwa dowolne wierzchołki (nadajniki) v i w mogą mieć ten sam kolor (przydzieloną tą samą częstotliwość) tylko wtedy gdy zachodzą dwa warunki:

1. Zarówno wierzchołek v nie leży w zasięgu wierzchołka w jak i odwrotnie, wierzchołek w jest poza zasięgiem nadajnika wierzchołka v.
2. Nie istnieje wierzchołek u taki, że jest on w zasięgu nadajników v i w.

Zauważmy, że problem optymalnego przydziału częstotliwości równoważny jest problemowi optymalnego kolorowania kwadratu grafu (patrz definicja 3.16). Z kolei ograniczając się tylko do pierwszego warunku, otrzymujemy klasyczny problem kolorowania grafu.

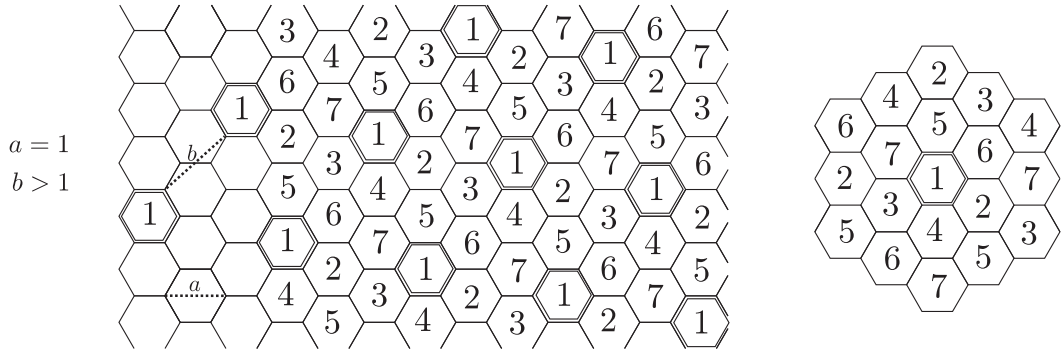
Rozpatrywany algorytm pracować będzie w rozproszonym modelu obliczeń LOCAL+COORD (definicja 1.3) na jednostkowych grafach dyskowych (definicja 1.2), w których każdy wierzchołek ma możliwość zmiany zasięgu swojego nadajnika.

W związku z tym dodatkowo dodajemy do grafu założenie, że każdy wierzchołek ma możliwość zmniejszenia promienia koła, które mu odpowiada. Wprowadźmy więc funkcję $f : V(G) \to (0, 1]$ przyporządkowującą wierzchołkowi, jego zredukowany promień. Wartość $f(v)$ możemy interpretować jako moc nadajnika wierzchołka v. Mając dany jednostkowy graf dyskowy G i funkcję $f : V(G) \to (0, 1]$ możemy zdefiniować skierowany graf $G\langle f\rangle$ w następujący sposób: $V(G\langle f\rangle) = V(G)$ i w $G\langle f\rangle$ krawędź łączy wierzchołek v z wierzchołkiem w wtedy i tylko wtedy gdy w jest zawarte w kole o promieniu $f(v)$ i środku w wierzchołku v. Formalnie $vw \in E(G\langle f\rangle) \Leftrightarrow \|v, w\| \leqslant f(v)$. Istnienie krawędzi skierowanej vw w $G\langle f\rangle$ możemy interpretować jako fakt, że v może wysłać wiadomość bezpośrednio do w (w jest w zasięgu komunikacji wierzchołka v).

Główny algorytm DiminishPower znajduje funkcję f i przydział częstotliwości w ten sposób, że wynikowy graf $G\langle f\rangle$ jest spójny, a liczb użytych częstotliwości jest znacząco mniejsza, od liczby chromatycznej pierwotnego grafu G.

Aby sformalizować wynik i przyszłe dowody wprowadźmy dwie definicje. Mówimy, że skierowany graf $G\langle f\rangle$ jest *silnie spójny*, jeśli dla wszystkich $v, w \in V(G\langle f\rangle)$ istnieje skierowana ścieżka z v do w. Silna spójność grafu $G\langle f\rangle$ implikuje możliwość komunikacji pomiędzy dowolnymi dwoma wierzchołkami w grafie $G\langle f\rangle$. Druga definicja formalizuje pojęcie sumarycznej energii zużywanej przez wszystkie nadajniki w sieci. Przez *koszt energetyczny* będziemy oznaczać wielkość $TE(G\langle f\rangle) = \sum_{v\in V(G\langle f\rangle)} f(v)^2$. Idea zmniejszania kosztu energetycznego w sieciach radiowych została opisana w pracach [63] i [49]. Podana tam definicja kosztu energetycznego sumuje wartości f w drugiej potędze. Wynika to z tego, że w przestrzeni dwuwymiarowej zasięg transmisji jest rzędu pierwiastka mocy nadajnika.

Główną ideą algorytmu DiminishPower jest podzielenie płaszczyzny na 7 klas heksagonów, przy czym każdy heksagon ma średnicę geometryczną 1 (patrz rysunek 4.6). Do każdej z klas przypiszmy inną paletę kolorów. Zauważmy, że podgraf G indukowany przez wierzchołki zawarte w wybranym heksagonie tworzy klikę. Ko-



Rysunek 4.6. Podział płaszczyzny na 7 klas heksagonów.

rzystając z tej obserwacji będziemy równolegle uruchamiać procedurę OnClique na każdym heksagonie. Procedura ta zmniejsza wartość funkcji f wewnątrz heksagonu i nadaje wierzchołkom z heksagonu kolory z odpowiedniej palety. Funkcja f jest dobrana wewnątrz heksagonu w ten sposób, że podgraf $G\langle f\rangle$ indukowany przez wierzchołki z heksagonu jest silnie spójny, a kolorowanie jest poprawne. Ostatnim krokiem algorytmu DiminishPower jest połączenie wszystkich heksagonów i poprawne kolorowanie grafu $G\langle f\rangle$.

Udowodnimy, że dla danego spójnego jednostkowego grafu dyskowego G algorytm DiminishPower znajduje w stałej liczbie rund synchronicznych funkcję $f : V(G) \to (0, 1]$ taką że:

1. $G\langle f\rangle$ jest silnie spójny (twierdzenie 4.11),
2. $G\langle f\rangle$ jest poprawnie pokolorowany przy pomocy $O(\log(\chi(G)))$ kolorów (twierdzenie 4.12),
3. Koszt energetycznych grafu $G\langle f\rangle$ jest rzędowo równy:

$$TE(G\langle f\rangle) = O\left(|MIS(G)| \log \frac{|V(G)|}{|MIS(G)|}\right)$$

gdzie $|MIS(G)|$ jest wielkością największego zbioru niezależnego w G (twierdzenie 4.3).

Dowód złożoności czasowej powyższej procedury zawarty jest w twierdzeniu 4.14.

Oprócz kolorowania przedstawimy także rozwiązanie problemu przydziału częstotliwości (wniosek 4.15). Pokażemy również jak przenieść protokół komunikacji (ang. routing protocol) z grafu G na graf $G\langle f\rangle$ (wniosek 4.16).

Porównajmy teraz nasz rezultat z wynikiem Fussen, Wattenhofer i Zollinger zawartym w pracy [24]. W [24] zaprezentowano algorytm NCC, znajdujący silne spójny graf $G_{ncc}\langle f\rangle$, czyli graf który spełnia warunek 1. W grafie $G_{ncc}\langle f\rangle$ maksymalny stopień jest rzędu $O(\log(|V(G)|))$, podczas gdy w grafie $G\langle f\rangle$ maksymalny stopien jest znacząco mniejszy i jest rzędu $O(\log(\omega(G)))$. W dodatku w pracy [24]

nie został rozwiązany problem przydziału częstotliwości i kolorowania jak również nie podano kosztu energetycznego. Algorytm NCC ściśle bazuje na z góry wybranym wierzchołku, tak zwanym korzeniu, a jego rozproszona implementacja zajmuje $O(diam(G))$ rund synchronicznych. Prezentowany przez nas algorytm jest znacznie szybszy i zajmuje $O(1)$ rund synchronicznych. Kolejną przewagą prezentowanego algorytmu jest to, że algorytm NCC nie pozwala zdefiniować szybkiego protokołu komunikacji (patrz dla porównania wniosek 4.16). Co więcej wierzchołki połączone krawędzią w grafie G mogą znajdować się w odległości $O(diam(G)\log(|V(G)|))$ od siebie w grafie $G_{ncc}\langle f\rangle$. Podczas gdy w grafie $G\langle f\rangle$ odległość ta jest znacząco mniejsza i wynosi $O(\log(\omega(G)))$.

Przystąpmy teraz do opisu naszego algorytmu. Najpierw zdefiniujmy procedurę OnClique która stanowi część składową głównego algorytmu DiminishPower. OnClique jednocześnie wykonywana jest na wszystkich podgrafach indukowanych przez heksagony. Ponieważ, jak już wspomniano, każdy heksagon indukuje klikę, zakładać więc możemy, że wejściowy graf K przedstawionego algorytmu jest kliką. Stąd też, informacja o całej klice może być wysłana do jednego wierzchołka, po czym całe lokalne obliczenia mogą być wykonywane w tym wierzchołku.

Na potrzeby algorytmu OnClique wprowadźmy dodatkową definicję. Mówimy, że vw jest *podwójnie skierowaną krawędzią*, jeśli w skierowanym grafie $G\langle f\rangle$ istnieją skierowane krawędzie z v do w i z w do v.

OnClique
Wejście: Jednostkowy graf dyskowy K będący kliką. Paleta $P(K)$ kolorów.
Wyjście: Graf dyskowy $K\langle f\rangle$ wraz z kolorowaniem.
(1) Niech $i := 1$, $M_i = V(K)$, $f \equiv 1$ i $K\langle f\rangle[M_i]$ będzie grafem indukowanym przez zbiór wierzchołków M_i.
(2) Dla wszystkich $v \in M_i$ ustaw $f(v) := \min_{w\in M_i, w\neq v} \|v, w\|$.
(3) Koloruj graf $K\langle f\rangle[M_i]$ używając 5 nowych kolorów z palety $P(K)$.
(4) Niech N będzie podgrafem $K\langle f\rangle[M_i]$ indukowanym przez wszystkie podwójnie skierowane krawędzie.
(5) $i := i + 1$.
(6) Oznacz przez M_i zbiór liderów składowych spójności grafu N.
(7) Jeśli $|M_i| \geqslant 2$ przejdź do kroku (2).
(8) Jeśli $M_i = \{v\}$ to $f(v) := 1$ i pokoloruj v nowym kolorem z palety $P(K)$.
(9) Zwróć $K\langle f\rangle[V(K)]$ wraz z kolorowaniem.

Należy podkreślić, że wykonanie kroku (3) procedury OnClique zawsze jest możliwe. Wynika to z lematu 4.9, który dowodzi, że graf $K\langle f\rangle[M_i]$ jest planarny.

W dodatku podkreślmy, że graf N jest zawsze dobrze zdefiniowany. Znaczy to, że ma co najmniej jeden wierzchołek, a w konsekwencji dla wszystkich i zbiór M_i zawiera co najmniej jeden element.

Aby udowodnić tą obserwację wystarczy zauważyć, że dla wszystkich M_i takich, że $|M_i| \geqslant 2$ i dla funkcji f zdefiniowanej w punkcie (2) graf $K\langle f\rangle[M_i]$ ma co najmniej jedną podwójnie skierowaną krawędź. Z definicji f w $K\langle f\rangle[M_i]$ każdy wierzchołek ma stopień wyjścia co najmniej jeden, stąd w $K\langle f\rangle[M_i]$ jest co najmniej jeden cykl, bądź krawędź podwójnie skierowana. Co więcej z definicji f każdy cykl w $K\langle f\rangle[M_i]$ zawiera tylko podwójnie skierowane krawędzie. Mówiąc precyzyjniej jeśli $v_1, \ldots, v_t, v_1$ jest cyklem, wtedy mamy: $f(v_1) = \min_{w\in M_i, w\neq v_1} \|v_1, w\| = \|v_1, v_2\| \geqslant f(v_2) = \|v_2, v_3\| \geqslant \ldots \geqslant f(v_t) = \|v_t, v_1\| \geqslant f(v_1)$, stąd wszystkie powyższe nierówności są równościami. Przez to wszystkie krawędzie w cyklu są podwójnie skierowane.

Wiedząc już, że algorytm OnClique jest dobrze określony, udowodnijmy kilka jego własności.

Lemat 4.7. *Niech $V(K) = M_1 \supseteq M_2 \supseteq M_3 \supseteq \ldots \supseteq M_k$ będzie ciągiem wszystkich podzbiorów zbioru $V(K)$ wygenerowanego przez algorytm* OnClique. *Wtedy dla każdego $1 \leqslant i \leqslant k$ zachodzi $|M_i| \geqslant 2\left|M_{(i+1)}\right|$ oraz $k \leqslant \log_2(|V(K)|)$ i $|M_k| = 1$.*

Dowód. Ponieważ N jest podgrafem $K\langle f\rangle[M_i]$ indukowanym przez pewne krawędzie, więc każda składowa spójności N zawiera co najmniej dwa wierzchołki. M_{i+1} z kolei zawiera dokładnie po jednym wierzchołku z każdej składowej spójności N. Stąd $|M_i| \geqslant 2\left|M_{(i+1)}\right|$ dla wszystkich $1 \leqslant i \leqslant k$. W związku z tym mamy $k \leqslant \log_2(|V(K)|)$. $|M_k| = 1$, ponieważ dla wszystkich i zbiór M_i zawiera co najmniej jeden element. □

Lemat 4.8. *Graf $K\langle f\rangle$ wygenerowany przez algorytm* OnClique *jest silnie spójny.*

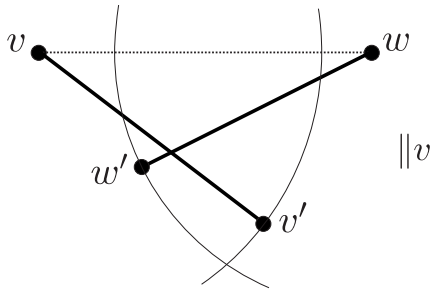
Dowód. Niech $V(K) = M_1 \supseteq M_2 \supseteq M_3 \supseteq \ldots \supseteq M_k = \{m\}$ będzie ciągiem podzbiorów $V(K)$ wygenerowanym przez algorytm OnClique. Niech $w = w_1$ będzie dowolnym wierzchołkiem z $V(K)$. Musimy pokazać, że w $K\langle f\rangle[V(K)]$ po ostatniej iteracji istnieje skierowana ścieżka z w do m. Łącząc to z obserwacją, że w $K\langle f\rangle$ są skierowane krawędzie łączące m ze wszystkimi innymi wierzchołkami (wynika to z faktu że $f(m) = 1$), otrzymamy silną spójność wynikowego grafu.

W pierwszej iteracji kroku (2) ustawmy $f(v) := \min_{w\in K, w\neq v} \|v, w\|$ dla wszystkich $v \in V(K) = M_1$. Z definicji $f(v)$ każdy wierzchołek v w $K\langle f\rangle[M_1]$ ma stopień wyjściowy co najmniej 1. Zauważmy, że implikuje to istnienie skierowanej ścieżki z $w = w_1$ do wierzchołka w_1', który jest incydentny do pewnej podwójnie skierowanej krawędzi. Wystarczy wziąć ostatni wierzchołek w najdłuższej skierowanej ścieżce zaczynającej się w w_1 i zauważyć że w_1' musi mieć sąsiada w tej ścieżce.

Stąd w_1' jest zawarty w skierowanym cyklu, a jak zauważyliśmy wcześniej cykl w $K\langle f\rangle[M_1]$ zawiera same podwójnie skierowane krawędzie. Jeśli w_2 jest liderem składowej spójności N, zdefiniowanym w kroku (4) podczas pierwszej iteracji, zawierającej w_1', wtedy istnieje skierowana ścieżka z w_1 do $w_2 \in M_2$.

Używając tego samego argumentu, możemy udowodnić, że istnieje skierowana ścieżka z $w_i \in M_i$ do $w_{i+1} \in M_{i+1}$ w $K\langle f\rangle[M_i]$ zdefiniowana w i-tej iteracji. Ponieważ $f(v)$ w następnych iteracjach może co najwyżej wzrastać, stąd połączenie ścieżek prowadzących z $w = w_1$ do w_2, z w_2 do w_3, ..., z w_{k-1} do $w_k = m$ tworzy skierowaną ścieżkę w wynikowym grafie $K\langle f\rangle[V(K)]$.

□



$$\|v,v'\| \leq \|v,w\| \text{ and } \|w,w'\| \leq \|v,w\|$$

Rysunek 4.7. $\|v,w'\| < \|v,v'\|$

Lemat 4.9. *Algorytm* OnClique *znajduje kolorowanie grafu* $K\langle f\rangle$ *używające co najwyżej* $5\log_2(|V(K)|)$ *kolorów z zadanej palety.*

Dowód. Najpierw zauważmy, że podczas i-tej iteracji algorytmu graf $K\langle f\rangle[M_i]$ jest planarny. W innym wypadku musiałyby istnieć w nim dwie krawędzi skierowane vv' i ww' przecinające się nawzajem. Tak więc z definicji f wynika, że $\|v,v'\| \leqslant \|v,w\|$ i $\|v,v'\| \leqslant \|v,w'\|$ oraz $\|w,w'\| \leqslant \|w,v\|$ i $\|w,w'\| \leqslant \|w,v'\|$. Jeśli założymy, że $\|v,v'\| \leqslant \|v,w\|$ i $\|w,w'\| \leqslant \|v,w\|$ wtedy w' będzie leżeć w kołu o środku v i promieniu równym $\|v,v'\|$ (patrz rysunek 4.7). Tak więc $\|v,w'\| < \|v,v'\|$ co stanowi sprzeczność.

Jak wiadomo możliwe jest pokolorowanie grafu planarnego przy użyciu 5 kolorów w czasie wielomianowym. Zatem w każdej iteracji będziemy używać co najwyżej 5 nowych kolorów z palety. Jeśli podczas iteracji zwiększymy wartość $f(v)$ wtedy w tej samej iteracji kolor v zmieni się. Będzie to jednak kolor inny od tych użytych w poprzednich iteracjach. Stąd wynikowe kolorowanie jest poprawnym kolorowaniem grafu $K\langle f\rangle$. Korzystając z lematu 4.7 uzyskujemy, że jest co najwyżej $\log_2(|V(K)|)$ iteracji, stąd kolorowanie używa co najwyżej $5\log_2(|V(K)|)$ kolorów. □

Przypomnijmy, że przez całkowity koszt energetyczny oznaczamy wielkość

$$TE(G\langle f\rangle) = \sum_{v\in V(G\langle f\rangle)} f(v)^2.$$

Lemat 4.10. *Dla grafu $K\langle f\rangle$, wygenerowanego przez algorytm* OnClique

$$TE(K\langle f\rangle) \leqslant 16\log_2(|V(K)|).$$

Dowód. Niech $V(K) = M_1 \supseteq M_2 \supseteq M_3 \supseteq \ldots \supseteq M_k$ będzie ciągiem podzbiorów $V(K)$ wygenerowanym przez algorytm OnClique. Niech $1 \leqslant i \leqslant k$ i $\left\{v_1, v_2, \ldots, v_{|M_i|}\right\} = M_i$. Podczas i-tej iteracji podstawiliśmy

$$f(v) := \min_{w\in M_i, w\neq v} \|v, w\| \text{ dla wszystkich } v \in M_i,$$

stąd $f(v_t) \leqslant \|v_t, v_l\|$ i $f(v_l) \leqslant \|v_t, v_l\|$ dla wszystkich $1 \leqslant k \neq l \leqslant |M_i|$. Implikuje to, że $f(v_t)/2 + f(v_l)/2 \leqslant \|v_t, v_l\|$ dla wszystkich $1 \leqslant k \neq l \leqslant |M_i|$. Tak więc, jeśli oznaczymy poprzez $c(v, r)$ koło o środku w v i promieniu długości r, wtedy wszystkie koła $c(v_1, f(v_1)/2)$, $c(v_2, f(v_2)/2)$, …, $c(v_{|M_i|}, f\left(v_{|M_i|}\right)/2)$ są parami rozłączne. Co więcej K jest kliką, stąd v_1 jest połączone krawędzią ze wszystkimi wierzchołkami w K. Ponieważ K jest jednostkowym grafem dyskowym, to wszystkie wierzchołki z K są zawarte w $c(v_1, 1)$. Implikuje to, że wszystkie koła $c(v_1, f(v_1)/2)$, $c(v_2, f(v_2)/2)$, …, $c(v_{|M_i|}, f\left(v_{|M_i|}\right)/2)$ leżą wewnątrz $c(v_1, 2)$. Stąd mamy

$$\pi 2^2 \geqslant \sum_{j=1}^{|M_i|} \pi\left(\frac{f(v_j)}{2}\right)^2.$$

Tak więc $TE(M_i) \leqslant \pi 2^2 \leqslant 16$.

Użyjmy tego samego rozumowania dla wszystkich iteracji i dla wszystkich wierzchołków $v \in V(K)$. Przypomnijmy, że wartość $f(v)$ jest liczona tylko w jednej iteracji. Tak więc, skoro z twierdzenia 4.7 $k \leqslant \log_2(|V(K)|)$, otrzymujemy, że

$$TE(K\langle f\rangle) \leqslant \sum_{j=0}^{k} TE(M_i) \leqslant \sum_{j=0}^{k} 16 = 16\log_2(|V(K)|).$$

□

Wprowadźmy teraz główny algorytm DiminishPower rozszerzający lokalne rozwiązania wygenerowane przez OnClique na cały graf. W algorytmie tym dla wszystkich $i = 1, 2, \ldots 7$, zdefiniujemy C_i jako zbiorów wszystkich składowych spójności podgrafu indukowanego przez wszystkie heksagony i-tej klasy (patrz rysunek 4.6). Naturalnie każda za tych składowych spójności jest kliką indukowaną przez pewne wierzchołki z heksagonu należącego do i-tej klasy. Algorytm DiminishPower przydzieli odpowiednim heksagonom rozłączne palety kolorów, wykona

DiminishPower
Wejście: Spójny jednostkowy graf dyskowy G
Wyjście: Graf dyskowy $G\langle f\rangle$ wraz z kolorowaniem.

(1) Podziel zbiór wierzchołków G na 7 klas heksagonów (jak na rysunku 4.6).
(2) Dla każdego $K \in C_i$ zdefiniuj paletę kolorów $P(K)$ jak następuje:

$$P(K) = \{x \in \mathbb{N} : x = i \pmod 7\}.$$

(3) Niech $\mathcal{C} := C_1 \cup C_2 \cup \ldots \cup C_7$. Dla każdego $K \in \mathcal{C}$ równolegle wykonaj:
   (a) Uruchom algorytm OnClique na grafie K i pokoloruj go używając palety $P(K)$.
   (b) Zdefiniuj $\mathcal{K}' := \{K' \in \mathcal{C} : K' \neq K \text{ oraz } \exists_{k\in V(K),k'\in V(K')} kk' \in E(G)\}$. Oznacz grafy ze zbioru $\mathcal{K}'$ jako $K'_1, K'_2, \ldots, K'_{|\mathcal{K}'|}$.
   (c) Dla kolejno $j = 0, 1, \ldots |\mathcal{K}'|$ wykonaj:
      (c1) Wybierz jeden wierzchołek $w_j \in V(K)$ taki, że $\exists_{w'_j\in V(K'_j)} w_j w'_j \in E(G)$.
      (c2) Pokoloruj w_j biorąc pierwszy kolor z palety $P(K)$
      (c3) Ustaw $f(w_j) = 1$.

równolegle procedurę OnClique, a następnie umożliwi komunikację pomiędzy sąsiednimi heksagonami.

Poniższe twierdzenia wskażą, że algorytm DiminishPower ma wszystkie własności, które postulowaliśmy.

Twierdzenie 4.11. *Mając dany spójny jednostkowy graf dyskowy G jako wejście, algorytm* DiminishPower *znajduje graf dyskowy $G\langle f\rangle$, który jest silnie spójny.*

Dowód. Niech $K, K' \in \mathcal{C}$. Korzystając z lematu 4.8 i z faktu, że podczas wykonywania DiminishPower w kroku 3(c) można tylko zwiększyć promień wierzchołków z $V(K)$ i $V(K')$, stąd w wynikowym grafie $G\langle f\rangle$ podgrafy indukowane przez wierzchołki z $V(K)$ i $V(K')$ są silnie spójne. Co więcej, jeśli w G były krawędzie pomiędzy $V(K)$ i $V(K')$, wtedy po kroku 3(c3) w $G\langle f\rangle$ jest co najmniej jedna krawędź skierowana z $V(K)$ do $V(K')$ i co najmniej jedna krawędź skierowana z $V(K')$ do $V(K)$. Stąd jeśli G był spójny, to $G\langle f\rangle$ jest silnie spójny. □

Twierdzenie 4.12. *Mając jako wejście jednostkowy graf dyskowy G, algorytm* DiminishPower *znajduje poprawne kolorowanie grafu $G\langle f\rangle$ używając $O(\log(\chi(G)))$ kolorów.*

Dowód. Z lematu 4.9 wynika, że w kroku (a) algorytmu DiminishPower, dla każdego $K \in \mathcal{C}$ użyjemy co najwyżej $5\log_2(|V(K)|)$ kolorów z palety $P(K)$. Co

więcej $|\mathcal{K}'| \leqslant 18$ (patrz rysunek 4.6), stąd dla dowolnego grafu $K \in \mathcal{C}$ w kroku 3(c2) użyjemy co najwyżej 18 kolorów z palety. Ponieważ jest dokładnie 7 palet oraz dla dowolnej kliki K zachodzi $|V(K)| \leqslant \omega(G)$, tak więc podczas całego algorytmu zostanie użytych co najwyżej $\max_{K\in\mathcal{C}} 7\,(5\log_2(|V(K)|) + 18) = O\,(\log(\omega(G)))$ kolorów. Rezultat ten wynika z faktu, że $\omega(G) \leqslant \chi(G) \leqslant 3\omega(G)$ (patrz praca [32]).

Korzystając z lematu 4.9 kolorowanie wygenerowane przez procedurę OnClique jest właściwym kolorowaniem wynikowego grafu. Co więcej w kroku 3(c), jeśli zmienimy promień koła odpowiadającego wierzchołkowi, to kolorujemy ten wierzchołek nowym kolorem, który nie był wcześniej użyty w K. Stąd, dla dowolnego $K \in \mathcal{C}$, kolorowanie wygenerowane przez algorytm DiminishPower jest poprawnym kolorowaniem grafu $G\langle f\rangle$, indukowanego przez wierzchołki z $V(K)$. Aby dowieść, że to kolorowanie jest również poprawnym kolorowaniem całego grafu $G\langle f\rangle$ wystarczy zauważyć, że w zasięgu komunikacji wierzchołków z K mieszczą się tylko wierzchołki z sąsiednich heksagonów (patrz rysunek 4.6). Wierzchołki te mają nadane odmienne od K palety, więc pokolorowane są innymi kolorami niż wierzchołkki z K. □

Twierdzenie 4.13. *Biorąc jako wejście spójny jednostkowy graf dyskowy G, algorytm* DiminishPower *znajduje graf $G\langle f\rangle$ taki, że całkowity koszt energetyczny $G\langle f\rangle$ jest równy:*

$$TE(G\langle f\rangle) = O\left(|MIS(G)|\log\left(\frac{|V(G)|}{|MIS(G)|}\right)\right),$$

gdzie $|MIS(G)|$ jest wielkością największego zbioru niezależnego w G.

Dowód. Dla każdego $K \in \mathcal{C}$, korzystając z lematu 4.10 i faktu, że w kroku 3(c) algorytmu DiminishPower powiększamy promień co najwyżej 18-tu wierzchołkom otrzymujemy, że dla funkcji f wygenerowanej przez algorytm DiminishPower mamy:

$$TE\left(G\langle f\rangle[V(K)]\right) \leqslant 16(\log_2(|V(K)|) + 18.$$

Stąd

$$\begin{aligned} TE(G\langle f\rangle) &\leqslant \sum_{K\in\mathcal{C}} [16(\log_2(|V(K)|) + 18] \\ &= 16\log_2\left(\prod_{K\in\mathcal{C}} |V(K)|\right) + 18|\mathcal{C}|. \end{aligned}$$

Zauważmy teraz, że $|MIS(G)| \leqslant |\mathcal{C}| \leqslant 7|MIS(G)|$. Wynika to z faktu, że w każdym $K \in \mathcal{C}$ jest co najwyżej jeden wierzchołek z $MIS(G)$ i możemy skonstruować niezależny zbiór wybierając po jednym wierzchołku z heksagonów z pierwszej klasy

podziału (patrz rysunek 4.6). Co więcej dla dowolnych liczb dodatnich a_i i liczby całkowitej n zachodzi: $((a_1 + a_2 + \ldots + a_n)/n)^n \geqslant a_1 a_2 \ldots a_n$. Stąd:

$$\begin{aligned} TE(G\langle f\rangle) &\leqslant 16\log_2\left(\frac{|V(G)|}{|\mathcal{C}|}\right)^{|\mathcal{C}|} + 18|\mathcal{C}| \\ &\leqslant 16\cdot 7|MIS(G)|\log_2\left(\frac{|V(G)|}{|MIS(G)|}\right) + 18|\mathcal{C}| \end{aligned}$$

□

Twierdzenie 4.14. *Algorytm* DiminishPower *może być zaimplementowany w rozproszonym modelu $LOCAL+COORD$ w stałej liczbie rund synchronicznych.*

Dowód. Ponieważ zakładamy, że każdy wierzchołek zna swoje współrzędne na płaszczyźnie, zatem krok (1) (czyli podzielenie na 7 klas heksagonów) i krok (2) (zdefiniowanie palet kolorów) algorytmu DiminishPower mogą być zaimplementowane w jednej rundzie synchronicznej. Dodatkowo każdy $K \in \mathcal{C}$ jest kliką, stąd informacja o całym grafie K może być wysłana do jednego wierzchołka lidera w K. Lider ten znając całą strukturę i współrzędne wierzchołków w swojej klice, lokalnie uruchamia algorytm OnClique. Stąd OnClique może być zaimplementowane w czterech rundach synchronicznych. Są nimi kolejno: wpierw wybieranie lidera, następnie wysyłanie mu całej informacji o heksagonie do którego należy, następnie lider lokalnie uruchamia algorytm OnClique w końcu lider wysyła wszystkim wierzchołkom z K wynik algorytmu. Ostatecznie $|\mathcal{K}'| \leqslant 18$, stąd krok (d) składa się z $O(1)$ rund synchronicznych. □

Jak już zostało wspomniane, przydział częstotliwości jest problemem równoważnym kolorowaniu kwadratu grafu. Aby otrzymać poprawny przydział częstotliwości zmodyfikujemy nieznacznie dwa kroki algorytmu DiminishPower.

— W kroku (1) algorytmu DiminishPower podziel zbiór wierzchołków w G na 12 (zamiast 7) heksagonów (patrz rysunek 2.2 lub rysunek 1 z pracy [35]).
— W kroku (3) algorytmu OnClique kolorujemy kwadrat grafu $K\langle f\rangle[M_i]$ używając 37 kolorów (zamiast 5) z palety $P(K)$, wykorzystując przy tym zachłanny algorytm kolorowania.

Obserwacja 4.15. *Algorytm* DiminishPower *z powyższymi zmianami tworzy w stałej liczbie synchronicznych rund graf $G\langle f\rangle$ z właściwym przydziałem częstotliwości. Co więcej używa on $O(\log(\chi(G)))$ kolorów i ma własności opisane w twierdzeniu 4.3 i twierdzeniu 4.11.*

Dowód. Po pierwsze, w podziale na 12 klas każde dwa wierzchołki z dwóch różnych heksagonów z tej samej klasy są od siebie odległe o więcej niż 2. Tak więc nie ma interferencji pomiędzy częstotliwościami wierzchołków z różnych heksagonów.

Po drugie, w i-tej iteracji algorytmu OnClique graf $K\langle f\rangle[M_i]$ ma maksymalny stopień ograniczony przez 6. Wynika to z obserwacji, że minimalny kąt pomiędzy krawędziami z grafie $K\langle f\rangle[M_i]$ jest $\frac{\pi}{3}$. By tego dowieść przypuśćmy, że istnieją dwie skierowane krawędzie $wv, w'v \in E(K\langle f\rangle[M_i])$ takie, że kąt $\angle(wvw') < \frac{\pi}{3}$ i $\|w,v\| \geqslant \|w',v\|$ wtedy $\|w,w'\| < \|w,v\|$. Otrzymujemy w ten sposób sprzeczność z założeniem, że $wv \in K\langle f\rangle[M_i]$. Stąd kwadrat grafu $K\langle f\rangle[M_i]$ ma stopień ograniczony przez 36. Ostatecznie używając algorytmu zachłannego można pokolorować graf $K\langle f\rangle[M_i]$ używając 37 kolorów.

Podsumowując, przydział częstotliwości wygenerowane przez zmodyfikowany algorytm DiminishPower używa co najwyżej $12\,(37\log_2(|\omega(G)|)+18)$ kolorów. □

Przedstawmy teraz w jaki sposób stworzyć protokół przekazywania informacji (ang. routing protocol) w wynikowym grafie $G\langle f\rangle$.

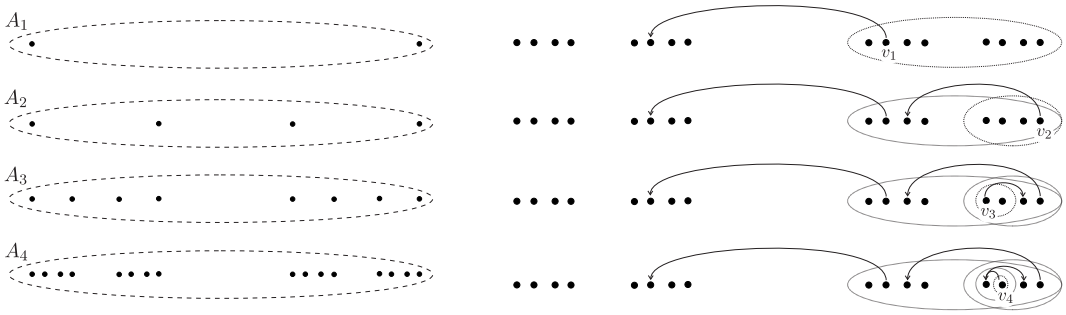
Obserwacja 4.16. *Jeśli mamy dany protokół komunikacyjny w jednostkowym grafie dyskowym G, wtedy możemy zdefiniować protokół komunikacyjny w grafie $G\langle f\rangle$. Dokładniej, jeśli protokół komunikacyjny w G używa krawędzi vw, wtedy by przekazać w grafie $G\langle f\rangle$ informację z v do w, użyjemy skierowanej ścieżki, której konstrukcja została pokazana w dowodzie twierdzenia 4.11. Precyzyjniej jako w'_i możemy wziąć pierwszy wierzchołek z skierowanej ścieżki zaczynającej się w w_i, incydentny do dwukrotnie skierowanej krawędzi w $G\langle f\rangle[M_i]$ zdefiniowanej w i-tej iteracji.*

Na zakończenie niniejszego rozdziału pokażemy, że liczba chromatyczna, oraz analogicznie przydział częstotliwości, grafu wygenerowanego przez algorytm DiminishPower jest optymalna z dokładnością do stałej. By to pokazać udowodnimy, że istnieje spójny jednostkowy graf dyskowy G taki, że dla dowolnej funkcji $f : V(G) \to (0,1]$ takiej, że $G\langle f\rangle$ jest silnie spójny, zachodzi $\chi(G\langle f\rangle) = \Omega(\log(\chi(G)))$.

Konstrukcja przykładowego grafu bazuje na zbiorze Cantora. Niech zbiór A_i będzie zbiorem punktów końcowych odcinków powstałych w i-tym kroku konstrukcji zbioru Cantora. Formalnie rzecz ujmując $A_1 = \{0,1\}$, $A_2 = \left\{0, \frac{1}{3}, \frac{2}{3}, 1\right\}$, $A_3 = \left\{0, \frac{1}{9}, \frac{2}{9}, \frac{1}{3}, \frac{2}{3}, \frac{7}{9}, \frac{8}{9}, 1\right\}$ itd., jak pokazano na rysunku 4.8. Niech G będzie jednostkowym grafem dyskowym ze zbiorem wierzchołków odpowiadającym punktom A_k. Jasnym jest, że G jest kliką i ma dokładnie 2^k wierzchołków.

Twierdzenie 4.17. *Niech G będzie jednostkowym grafem dyskowym o zbiorze wierzchołków A_k i $f : V(G) \to (0,1]$ będzie dowolną funkcją. Jeśli $G\langle f\rangle$ jest silnie spójny wtedy, $\chi(G\langle f\rangle) = \Omega(\log(\chi(G)))$.*

Dowód. Zdefiniujmy najpierw $G\{x_1, x_2\}$ jako podzbiór wierzchołków grafu G ze współrzędną x pomiędzy x_1 a x_2. Ponieważ $G\langle f\rangle$ jest spójny, więc istnieje co najmniej jedna krawędź skierowana e_1 z $v_1 \in G\langle f\rangle\{\frac{2}{3}, 1\}$ do $w_1 \in G\langle f\rangle\{0, \frac{1}{3}\}$. Bez straty ogólności przyjmijmy, że $v_1 \in G\langle f\rangle\{\frac{2}{3}, \frac{7}{9}\}$ ma kolor 1 (patrz rysunek 4.8). Zauważmy, że skoro $f(v_1) \geqslant \frac{1}{3}$ to wszystkie pozostałe wierzchołki z $G\langle f\rangle\{\frac{2}{3}, 1\}$ nie mogą mieć koloru 1. Analogicznie, istnieje co najmniej jedna skierowana krawędź e_2 z $v_2 \in G\langle f\rangle\{\frac{8}{9}, 1\}$ do $w_2 \in G\langle f\rangle\{\frac{2}{3}, \frac{7}{9}\}$. Bez straty ogólności przyjmijmy, że $v_2 \in G\langle f\rangle\{\frac{26}{27}, 1\}$ i ma kolor 2 (patrz rysunek 4.8). Zauważmy teraz, że skoro $f(v_2) \geqslant \frac{1}{9}$ to wszystkie inne wierzchołki z $G\langle f\rangle\{\frac{8}{9}, 1\}$ nie mogą mieć koloru 2. Kontynuując to rozumowanie, w kroku k-tym musimy użyć co najmniej k kolorów by poprawnie pokolorować $G\langle f\rangle$. Ponieważ G jest kliką $\chi(G) = V(G) = 2^k$, tak więc $\chi(G\langle f\rangle) \geqslant \log_2(\chi(G))$. □



Rysunek 4.8. Graf dla którego $\chi(G\langle f\rangle) = \Omega(\log(\chi(G)))$

Bibliografia

[1] A. D. Amis, R. Prakash, D. Huynh, i T. Vuong:, *Max-min d-cluster formation in wireless ad hoc networks*, Proceedings IEEE INFOCOM 2000, 2000, pp. 32–41.

[2] S. Arya i M. Smid, *Efficient construction of a bounded-degree spanner with low weight*, Algorithmica **17** (1997), no. 1, 33–54.

[3] B. Awerbuch, A. V. Goldberg, M. Luby, i S. A. Plotkin, *Network decomposition and locality in distributed computation*, 30th Annual Symposium on Foundations of Computer Science (FOCS 1989), IEEE Computer Society, 1989, pp. 364–369.

[4] S. Banerjee i S. Khuller, *A clustering scheme for hierarchical routing in wireless networks*, Proceedings IEEE INFOCOM 2001, The Conference on Computer Communications, vol. 2, 2001, pp. 1028–1037.

[5] L. Blin i F. Butelle, *A very fast (linear time) distributed algorithm, on general graphs, for the minimum-weight spanning tree.*, Procedings of the 5th International Conference on Principles of Distributed Systems. OPODIS 2001, 2001, pp. 113–124.

[6] M. Burkhart, P. von Rickenbach, R. Wattenhofer, i A. Zollinger, *Does topology control reduce interference?*, Proceedings of the 5th ACM Interational Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2004, 2004, pp. 9–19.

[7] M. Cardei, J. Wu, i S. Yang, *Topology control in ad hoc wireless networks using cooperative communication*, IEEE Transactions on Mobile Computing **5** (2006), no. 6, 711–724.

[8] Y. P. Chen, A. L. Liestman, i J. Liu, *Ad hoc and sensor networks: Wireless networks and mobile computing (vol. 2)*, ch. Clustering algorithms for ad hoc wireless networks, pp. 145–164, Nova Science Publisher, 2006.

[9] F. Y. L. Chin i H. F. Ting, *An almost linear time and o(n log n + e) messages distributed algorithm for minimum-weight spanning trees*, Proceedings of the 26th Annual Symposium on Foundations of Computer Science (FOCS 1985), 1985, pp. 257–266.

[10] I. Chlamtac i S. Kutten, *On broadcasting in radio networks - problem analysis and protocol design*, IEEE Transactions on Communications **33** (1985), 12401246.

[11] I. Chlamtac i O. Weinstein, *The wave expansion approach to broadcasting in multihop radio networks*, IEEE Trans. on Communications **39** (1991), 26–433.

[12] M. Chrobak, L. Gasieniec, i W. Rytter, *Fast broadcasting and gossiping in radio networks*, FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science (Washington, DC, USA), IEEE Computer Society, 2000, p. 575.

[13] M. Chrobak, L. Gasieniec, i W. Rytter, *A randomized algorithm for gossiping in radio networks*, COCOON '01: Proceedings of the 7th Annual International Con-

ference on Computing and Combinatorics (London, UK), Springer-Verlag, 2001, pp. 483–492.

[14] M. Chrobak, L. Gasieniec, i W. Rytter, *Fast broadcasting and gossiping in radio networks*, J. Algorithms **43** (2002), no. 2, 177–189.

[15] M. Couture, M. Barbeau, P. Bose, P. Carmi, i E. Kranakis, *Location oblivious distributed unit disk graph coloring.*, SIROCCO (Giuseppe Prencipe i Shmuel Zaks, eds.), Lecture Notes in Computer Science, vol. 4474, Springer, 2007, pp. 222–233.

[16] A. Czygrinow i M. Hańćkowiak, *Distributed approximation algorithms in unit-disk graphs*, Distributed Computing, LNCS 4167, 2006, pp. 385–398.

[17] J. Czyzowicz, S. Dobrev, T. Fevens, H. Gonzlez-Aguilar, E. Kranakis, J. Opatrny, i J. Urrutia, *Local algorithms for dominating and connected dominating sets of unit disk graphs with location aware nodes.*, LATIN, Lecture Notes in Computer Science, vol. 4957, Springer, 2008, pp. 158–169.

[18] A. Dessmark i A. Pelc, *Tradeoffs between knowledge and time of communication in geometric radio networks*, SPAA '01: Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures (New York, NY, USA), ACM, 2001, pp. 59–66.

[19] I. I. Er i W. K. G. Seah, *Performance analysis of mobility-based d-hop (mobdhop) clustering algorithm for mobile ad hoc networks*, Comput. Netw. **50** (2006), no. 17, 3375–3399.

[20] T. Erlebach i J. Fiala, *Independence and coloring problems on intersection graphs of disks*, Efficient Approximation and Online Algorithms **Volume 3484** (2006), 135–155.

[21] F. Manne F. Cicalese i Q. Xin, *Faster centralized communication in radio networks*, LNCS **4288** (2006), 339–348.

[22] M. Faloutsos i M. Molle, *Creating optimal distributed algorithms for minimum spanning trees*, 1995.

[23] Y. Fernandess i D. Malkhi, *K-clustering in wireless ad hoc networks*, POMC '02: Proceedings of the second ACM international workshop on Principles of mobile computing (New York, NY, USA), ACM, 2002, pp. 31–37.

[24] M. Fussen, R. Wattenhofer, i A.Zollinger, *Interference Arises at the Receiver*, International Conference on Wireless Networks, Communications, and Mobile Computing (WIRELESSCOM), Maui, Hawaii, USA, June 2005.

[25] I. Gaber i Y. Mansour, *Centralized broadcast in multihop radio networks.*, Journal of Algorithms **46** (2003), 120.

[26] R. G. Gallager, P. A. Humblet, i P. M. Spira, *A distributed algorithm for minimum-weight spanning trees*, ACM Trans. Program. Lang. Syst. **5** (1983), no. 1, 66–77.

[27] R. Gandhi, S. Parthasarathy, i A. Mishra, *Minimizing broadcast latency and redundancy in ad hoc networks*, MobiHoc '03: Proceedings of the 4th ACM international symposium on Mobile ad hoc networking & computing (New York, NY, USA), ACM, 2003, pp. 222–232.

[28] J. A. Garay, S. Kutten, i D. Peleg, *A sub-linear time distributed algorithm for*

minimum-weight spanning trees (extended abstract), IEEE Symposium on Foundations of Computer Science, 1993, pp. 659–668.

[29] L. Gąsieniec i A. Lingas, *On adaptive deterministic gossiping in ad hoc radio networks*, Inf. Process. Lett. **83** (2002), no. 2, 89–93.

[30] L. Gąsieniec i I. Potapov, *Gossiping with unit messages in known radio networks*, TCS '02: Proceedings of the IFIP 17th World Computer Congress - TC1 Stream / 2nd IFIP International Conference on Theoretical Computer Science (Deventer, The Netherlands, The Netherlands), Kluwer, B.V., 2002, pp. 193–205.

[31] A. V. Goldberg, S. A. Plotkin, i G. E. Shannon, *Parallel symmetry-breaking in sparse graphs*, SIAM J. Disc. Math, 1987, pp. 315–324.

[32] A. Gräf, M. Stumpf, i G. Weißenfels, *On coloring unit disk graphs*, Algorithmica **20** (1998), 277–293.

[33] T.-C. Hou i V. Li, *Transmission range control in multihop packet radio networks*, Communications, IEEE Transactions on [legacy, pre - 1988] **34** (1986), no. 1, 38–44.

[34] S. C.-H. Huang, H. Du, i E.-K. Park, *Minimum-latency gossiping in multi-hop wireless networks*, MobiHoc '08: Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing (New York, NY, USA), ACM, 2008, pp. 323–330.

[35] S. C. H. Huang, P.-J. Wan, X. Jia, i H. Du, *Low-latency broadcast scheduling in ad hoc networks*, Lecture Notes in Computer Science **4138** (2006), 527–538.

[36] S. C.-H. Huang, P.-J. Wan, X. Jia, i H. Du, *Minimum-latency broadcast scheduling in wireless ad hoc networks*, INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE, 2007, pp. 733–739.

[37] D. Kowalski i A. Pelc, *Centralized deterministic broadcasting in undirected multi-hop radio network*, In Random-Approx (2004), 171182.

[38] D. Kowalski i A. Pelc, *Optimal deterministic broadcasting in known topology radio networks*, Distributed Computing **19(3** (2007), 185–195.

[39] M. Krunz, A. Muqattash, i S.-J. Lee, *Transmission power control in wireless ad hoc networks: challenges, solutions, and open issues*, IEEE Network **18** (2004), 8–14.

[40] K. Krzywdziński, *Distributed channel assignment in sensor networks with diminished transmitter power*, złożona do 17th International Colloquium on Structural Information and Communication Complexity SIROCCO 2010.

[41] K. Krzywdziński, *Distributed construction of broadcast scheduling and gossiping in dynamic radio networks*, złożona do 17th International Colloquium on Structural Information and Communication Complexity SIROCCO 2010.

[42] K. Krzywdziński, *Efficient construction of (d+1,3d)-ruling set in wireless ad hoc networks*, Challenges in Computational Collective Intelligence (Adam Janiak Ngoc Thanh Nguyen, Radoslaw Katarzyniak, ed.), Springer Series Studies in Computational Intelligence, 2009.

[43] K. Krzywdziński, *Distributed algorithm finding regular clustering in unit disc graphs*, Proceedings of 36th International Conference on Current Trends in Theory and Practice of Computer Science SOFSEM, 2010.

[44] K. Krzywdziński, *A local distributed algorithm to approximate mst in unit disc graphs*, Lecture Notes in Computer Science 5699, FCT 2009.

[45] F. Kuhn, T. Moscibroda, i R. Wattenhofer, *On the locality of bounded growth*, PODC '05: Proceedings of the twenty-fourth annual ACM symposium on Principles of distributed computing (New York, NY, USA), ACM, 2005, pp. 60–68.

[46] D.Peleg L.Gasieniec i Q.Xin, *Faster communication in known topology radio networks*, Proceedings of The 24th Annual ACM Symposium on Principles of Distributed Computing, 2005.

[47] N. Li, J. C. Hou, i L. Sha, *Design and analysis of an mst-based topology control algorithm*, INFOCOM, 2003.

[48] X.-Y. Li, *Approximate mst for udg locally*, (COCOON 2003,) Lecture Notes in Computer Science **2697** (2003), 364–373.

[49] X.-Y. Li i P.-J. Wan, *Minimum energy mobile wireless networks revisited*, In Proc. IEEE International Conference on Communications (ICC, 2001, pp. 278–283.

[50] K. M. Lillis, S. V. Pemmaraju, i I. A. Pirwani, *Topology control and geographic routing in realistic wireless networks*, ADHOC-NOW, 2007, pp. 15–31.

[51] C. R. Lin i M. Gerla, *Adaptive clustering for mobile wireless networks*, IEEE Journal on Selected Areas in Communications **15** (1997), 1265–1275.

[52] N. Linial i M. Saks, *Decomposing graphs into regions of small diameter*, SODA '91: Proceedings of the second annual ACM-SIAM symposium on Discrete algorithms (Philadelphia, PA, USA), Society for Industrial and Applied Mathematics, 1991, pp. 320–330.

[53] G. Kortsarz M. Elkin, *Improved broadcast schedule for radio networks*, Symposium on Discrete Algorithms (SODA), 2005, 222-231.

[54] G. Pandurangan M. Khan i V.S.A. Kumar, *Distributed algorithms for constructing approximate minimum spanning trees in wireless sensor networks*, IEEE Transactions on Parallel and Distributed Systems, to appear.

[55] K. Moaveni-Nejad i X.-Y. Li, *Low-interference topology control for wireless ad hoc networks*, ACM Wireless Networks, IEEE Press, 2005.

[56] A. Panconesi i A. Srinivasan, *On the complexity of distributed network decomposition*, J. Algorithms **20** (1996), no. 2, 356–374.

[57] S. Parthasarathy i R. Gandhi, *Distributed algorithms for coloring and domination in wireless ad hoc networks*, FSTTCS 2004: Foundations of Software Technology and Theoretical Computer Science, LNCS 3328, 2004, pp. 447–459.

[58] R. Peeters, *On coloring j-unit sphere graphs*, Tech. report, Dept. of Economics, Tilburg Universit, 1991.

[59] D. Peleg, *Distributed computing: a locality-sensitive approach*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[60] D. Peleg i V. Rubinovich, *A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction*, SIAM J. Comput. **30** (2000), no. 5, 1427–1442.

[61] L. Ramachandran, M. Kapoor, A. Sarkar, i A. Aggarwal, *Clustering algorithms for wireless ad hoc networks*, DIALM '00: Proceedings of the 4th international workshop

on Discrete algorithms and methods for mobile computing and communications (New York, NY, USA), ACM, 2000, pp. 54–63.

[62] R. Ramanathan i R. Rosales-Hain, *Topology control of multihop wireless networks using transmit power adjustment*, Proc. of the 19 th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2000.

[63] V. Rodoplu i T. H. Meng, *Minimum energy mobile wireless networks*, Selected Areas in Communications, IEEE Journal on **17** (1999), no. 8, 1333–1344.

[64] J. Schneider i R. Wattenhofer, *A log-star distributed maximal independent set algorithm for growth-bounded graphs*, 27th ACM Symposium on Principles of Distributed Computing (PODC), Toronto, Canada, August 2008.

[65] H. Takagi i L. Kleinrock, *Optimal transmission ranges for randomly distributed packet radio terminals*, Communications, IEEE Transactions on [legacy, pre - 1988] **32** (1984), no. 3, 246–257.

[66] S.-C. Wang, C.-Y. Lin, i S.-Y. Kuo, *A localized topology control algorithm for constructing power efficient wireless ad hoc networks*, World Wide Web Conference Series, 2003.

[67] R. Wattenhofer, L. Li, P. Bahl, i Y. Wang, *Distributed topology control for power efficient operation in multihop wireless ad hoc networks*, IEEE INFOCOM, 2001, pp. 1388–1397.