Autoreferat

Autor: Michał Hańćkowiak

Tytuł rozprawy: „Rozproszone algorytmy aproksymacyjne”

1 Dyplomy, stopnie naukowe oraz zatrudnienie.

- 1993, stopień magistra
- 2001, stopień doktora nauk matematycznych w zakresie informatyki, tytuł rozprawy doktorskiej: „Algorytmy rozproszone poszukiwania maksymalnych skojarzeń w grafach”, promotor: prof. Michał Karoński, recenzenci: prof. Bogdan Chlebus, prof. Mirosław Kutyłowski
- 1.10.1993, zatrudnienie na Wydziale Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu, zakład Matematyki Dyskretnej, potem Zakład Teorii Algorytmów i Bezpieczeństwa Danych
- 1996-2000, oddelegowany na studia doktoranckie
- od 1.06.2001 adiunkt

2 Działalność naukowa.

2.1 Liczba prac.

Łączna liczba opublikowanych prac: 18, z czego 3 przed doktoratem, 3 prace czasopismowe.

2.2 Udział w konferencjach.

Wyglosiłem referaty na następujących, międzynarodowych konferencjach:

- ESA 2001, 9th Annual European Symposium On Algorithms, Aarhus, Denmark, August 28-31, 2001,
- COCOON 2003, The Ninth International Computing and Combinatorics Conference, July 25-28, 2003, Big Sky, MT, USA,

- CIAC 2006, Algorithms and Complexity, 6th Italian Conference, Rome, Italy, May 29-31, 2006,
- DISC 2007, Distributed Computing, 21st International Symposium, Lemesos, Cyprus, September 24-26, 2007,
- DISC 2008, Distributed Computing, 22nd International Symposium, Arcachon, France, September 22-24, 2008„
- DISC2012, Distributed Computing, 26th International Symposium, Salvador, Brazil, October 16-18, 2012.

2.3 Udział w grantach.

- „Rozproszone algorytmy grafowe", 7 T11C 032 20, wykonawca,
- „Rozproszona aproksymacja problemów optymalizacyjnych w sieciach", N206 017 32/2452, kierownik,
- „Algorytmiczne aspekty teorii grafów i hipergrafów w klasycznym i rozproszonym modelu obliczeń", N N206 565740, wykonawca.

2.4 Nagrody.

Nagroda indywidualna 3 stopnia za osiągnięcia w pracy naukowej.

2.5 Osiągnięcia dydaktyczne.

Byłem opiekunem naukowym Wojciecha Wawrzyniaka, gdy przygotowywał rozprawę doktorską zatytułowaną „Rozproszone algorytmy aproksymacjne w analizie własności grafowych".

Wypromowałem 21 magistrów, tematyka prac magisterskich: informatyka, zarówno w ujęciu teoretycznym jak i stosowanym. Prowadzę lub prowadziłem zajęcia dydaktyczne z przedmiotów informatyki teoretycznej, np. „Algorytmy rozproszone", „Algorytmy i struktury danych", jak i informatyki stosowanej, np. „Systemy operacyjne", „Technologie aplikacji serwerowych", „Projekt zespołowy", „Sieci komputerowe".

3 Osiągnięcie naukowe, o którym mowa w art. 16 ust. 2 ustawy o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki (Dz. U. nr 65, poz. 595 ze zm.)

Powyższym osiągnięciem naukowym jest jednotematyczny cykl 9 publikacji zatytułowany „Rozproszone algorytmy aproksymacyjne". Publikacje te są wymienione poniżej i oznaczone przez **[h1]**, ..., **[h9]**.

[h1] A. Czygrinow, M. Hańćkowiak: Distributed algorithms for weighted problems in sparse graphs, J. Discrete Algorithms 4(4), 2006, pp. 588-607,

[h2] A. Czygrinow, M. Hańćkowiak, E. Szymanska: Distributed Approximation Algorithms for Planar Graphs, CIAC 2006, Algorithms and Complexity, 6th Italian Conference, Rome, Italy, May 29-31, 2006, pp. 296-307,

[h3] A. Czygrinow, M. Hańćkowiak: Distributed Almost Exact Approximations for Minor-Closed Families, ESA 2006, Algorithms, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, pp. 244-255,

[h4] A. Czygrinow, M. Hańćkowiak: Distributed Approximation Algorithms for Weighted Problems in Minor-Closed Families, COCOON 2007, Computing and Combinatorics, 13th Annual International Conference, Banff, Canada, July 16-19, 2007, pp. 515-525,

[h5] A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Distributed packing in planar graphs, SPAA 2008, Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Munich, Germany, June 14-16, 2008, pp. 55-61,

[h6] A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Fast Distributed Approximations in Planar Graphs, DISC 2008, Distributed Computing, 22nd International Symposium, Arcachon, France, September 22-24, 2008, pp. 78-92,

[h7] E. Szymanska, A. Czygrinow, M. Hańćkowiak: Fast Distributed Approximation Algorithm for the Maximum Matching Problem in Bounded Arboricity Graphs, ISAAC 2009, Algorithms and Computation, 20th International Symposium, Honolulu, Hawaii, USA, December 16-18, 2009, pp. 668-678,

[h8] E. Szymanska, K. Krzywdzinski, A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Distributed Approximation Algorithm for the Semi-Matching Problem, DISC 2011 (Brief

Announcements only !), Distributed Computing, 25th International Symposium, Rome, Italy, September 20-22, 2011, pp. 200-201,

[h9] E. Szymanska, A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Distributed 2-approximation Algorithm for the Semi-Matching Problem, DISC 2012, Distributed Computing, 26th International Symposium, Salvador, Brazil, October 16-18, 2012, pp. 210-222.

4 Wprowadzenie w tematykę rozprawy.

Moja rozprawa habilitacyjna, w skład której wchodzą wymienione wyżej prace [**h1**]-[**h9**], poświęcona jest optymalizacyjnym problemom grafowym w rozproszonym, synchronicznym modelu obliczeń.

Systemy rozproszone występują w wielu miejscach i na różnych poziomach w otaczającym nas świecie. Jako przykłady można wymienić: sieci telekomunikacyjne, Internet, aplikacje rozproszone używające wirtualnych połączeń (tcp), mózg żywego zwierzęcia, podzespoły komputera komunikujące się poprzez magistrale. Główną cechą systemu rozproszonego jest to, że składa się on z elementów (procesorów), które nie „widzą" całego systemu, a jedynie jego część, a mimo to muszą rozwiązywać problemy globalne; np. jeśli wierzchołek A chce wysłać wiadomość do wierzchołka B to może być zainteresowany znalezieniem najkrótszej ścieżki z A do B (to jest przykład problemu globalnego). Aby rozwiązać problem globalny elementy systemu komunikują się z innymi elementami wysyłając i odbierając komunikaty.

System rozproszony opisujemy przy pomocy *modelu rozproszonego*. Model rozproszony składa się z *grafu komunikacyjnego*, którego wierzchołki to procesory, a krawędzie służą do przesyłania komunikatów. Wierzchołek może się bezpośrednio komunikować wyłącznie z sąsiadami w grafie komunikacyjnym. Zakłada się, że wierzchołki są wyposażone w (różne) identyfikatory. Rozróżnia się przynajmniej dwa rodzaje modeli rozproszonych, *synchroniczny* i *asynchroniczny*, przy czym ten pierwszy zakłada istnienie globalnego zegara pozwalającego wykonywać wszystkim procesorom pojedynczy krok obliczeniowy równocześnie. W każdym takim kroku każdy wierzchołek wysyła i odbiera komunikaty od sąsiadów oraz wykonuje lokalne obliczenia. Efektywność algorytmów rozproszonych mierzymy przy pomocy ich złożoności komunikatowej (liczby wysyłanych komunikatów) oraz złożoności czasowej (w modelu synchronicznym jest to liczba kroków obliczeniowych, tzw *rund*).

Początkowo, w latach 70/80, większy nacisk kładziono na model rozproszony asynchroniczny, jako bardziej odpowiadający naturze sieci komputerowych. W modelu tym rozważano elementarne problemy takie jak *wybór lidera* w cyklu lub w grafie pełnym [2],

[3]. Chyba jedynym rozważanym wtedy problemem grafowym był problem MST (Minimum Spanning Tree) [4]. W powyższych algorytmach analizowano głównie złożoność komunikatową. Zauważono także, że projektowanie algorytmów w modelu asynchronicznym jest dużo trudniejsze niż w modelu synchronicznym, stąd pojawiły się narzędzia pozwalające uruchamiać algorytmy synchroniczne w sieci asynchronicznej, tzw. *synchronizatory*, np. synchronizatory α, β, γ [1].

Na przełomie lat 80/90 zwiększyło się zainteresowanie problemami grafowymi w modelu synchronicznym zapoczątkowane pracą Liniala [11]. Praca ta omawiała problem MIS (Maximal Independent Set) w cyklu. MIS to maksymalny w sensie zawierania zbiór niezależnych (czyli niepołączonych krawędzią) wierzchołków w grafie. Praca [11] zawiera dowód, że rozwiązanie tego problemu wymaga czasu $\Omega(\log^* n)$. Równocześnie znany jest algorytm znajdujący MIS w cyklu w czasie $O(\log^* n)$, wykorzystujący technikę Cole/Vishkina [9], używającą w nietrywialny sposób identyfikatorów wierzchołków. Nadal pozostaje otwarta kwestia możliwości deterministycznego obliczania MIS w dowolnym grafie, w czasie polilogarytmicznym. (Znany jest natomiast algorytm obliczający MM (Maximal Matching), działający w czasie $O(\log^4 n)$ [25], będący częścią mojej pracy doktorskiej.)

W książce Pelega [16] z roku 2000 pojawiła się nazwa LOCAL dla modelu synchronicznego z dodatkowym wymaganiem, aby czas działania algorytmów był krótki. Jeśli czas ten jest stały to mówimy wtedy o modelu „ściśle lokalnym". Jeśli jest funkcją zależną od n, to powinna to być funkcja o małych wartościach, np. polilogarytmiczna lub jeszcze mniejsza. Długość komunikatów i czas obliczeń lokalnych, tj. wykonywanych przez pojedynczy procesor, mogą być w modelu LOCAL nieograniczone. Pojawiła się też odmiana tego modelu o nazwie CONGEST, w której dodatkowo zakłada się, że komunikaty są długości $O(\log n)$. W tym okresie zaczęto rozpatrywać także inne niż MIS problemy grafowe, takie jak MM (Maximal Matching), MDS (Minimum Dominating Set) i inne.

Jaka jest motywacja do rozpatrywania problemów grafowych w modelu LOCAL? Na pewno można tu wymienić zapotrzebowanie na struktury danych w innych algorytmach. Część problemów ma naturalną motywację, np. minimalny zbiór dominujący (MDS) wyznacza optymalne rozmieszczenie serwerów w sieci. Zauważmy też, że algorytmy grafowe niekoniecznie muszą być uruchamiane w (fizycznym) grafie komunikacyjnym, ale mogą też być uruchamiane w rozmaitych grafach wirtualnych. Oznacza to, że dysponując algorytmem obliczającym MIS możemy w rzeczywistości rozwiązać wiele innych problemów, np. jeśli potrzebujemy wierzchołków, z których każde dwa są w odległości większej niż $k+1$, to wystarczy uruchomić algorytm znajdujący MIS w grafie (wirtualnym) G^k, który powstaje z grafu G przez dodanie (wirtualnej) krawędzi między wierzchołkami w odległości $\leq k$.

W modelu rozproszonym problemy optymalizacyjne zazwyczaj nie mogą być rozwiązy-

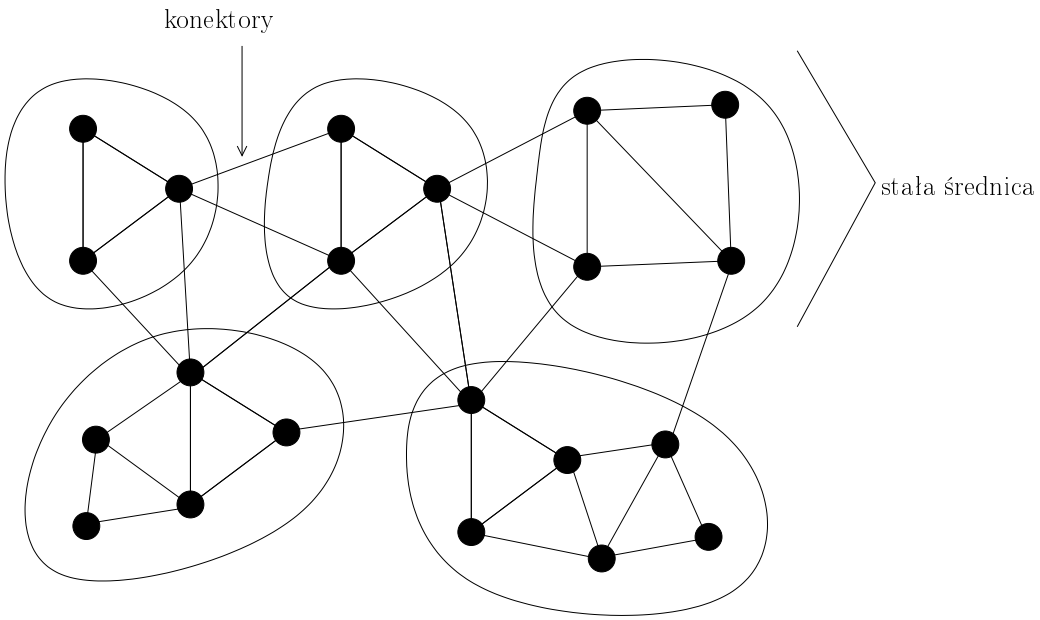
wane dokładnie dlatego poszukuje się algorytmów „aproksymacyjnych" znajdujących przybliżone rozwiązania. Jakość aproksymacji mierzona jest tzw. *współczynnikiem aproksymacji*. W przypadku problemu MIS, jeśli I^* oznacza optymalne rozwiązanie problemu, a I jest zbiorem wierzchołków obliczanym przez nasz algorytm, to współczynnikiem aproksymacji nazywamy dolne ograniczenie liczby $|I|/|I^*|$. Rozważa się algorytmy znajdujące rozwiązania ze stałym współczynnikiem aproksymacji, ze współczynnikiem $1-\epsilon$ dla dowolnego stałego ϵ (jest to aproksymacja typu PTAS [1]), lub ze współczynnikiem $1-f(n)$ gdzie $f(n) \to 0$ (aproksymacja typu FPTAS [2]). a także znajdujące rozwiązania maksymalne „w sensie zawierania", (w których nazwie używa się słowa maximal zamiast maximum).

W latach 90 pojawiło się narzędzie ogólnego przeznaczenia pozwalające rozwiązywać różne problemy grafowe: dekompozycja Liniala/Saksa [12], [13]. (D, C)-decompozycja Liniala/Saksa to podział zbioru wierzchołków grafu na podzbiory $V_1, \ldots, V_k$ zwane „klastrami", indukujące grafy o średnicy D. Klastry posiadają kolory ze zbioru mocy C przypisane w taki sposób, że klastry połączone krawędzią mają różne kolory. Posiadając (D, C)-dekompozycję w grafie komunikacyjnym z maksymalnym stopniem $\Delta(G)$ możemy w nim obliczyć MIS (typu Maximal), a także $(\Delta(G)+1)$-kolorowanie wierzchołkowe, w czasie $O(D \times C)$. W pierwszej z wymienionych prac podano algorytm losowy znajdujący $(\log n, \log n)$-dekompozycję, w oczekiwanym czasie $O(\log^2 n)$, natomiast w drugiej podano algorytm deterministyczny znajdujący taką samą dekompozycję, ale już nie w czasie polilogarytmicznym (gdyby tak było to istniałby polilogarytmiczny algorytm dla problemu MIS). Podział grafu na klastry jest też podstawowym narzędziem używanym w algorytmach aproksymacyjnch opisanych w pracach [**h1**]-[**h6**].

Kolejnym krokiem, jeśli chodzi o rozwiązywanie optymalizacyjnych problemów grafowych w modelu rozproszonym, były algorytmy wykorzystujące techniki programowania liniowego (LP, Linear Programming). W pracach grupy szwajcarskiej, Wattenhofera, Kuhna, Moscibrody i innych, [6], [7], przedstawiono liczne algorytmy znajdujące aproksymację klasycznych problemów grafowych, działające w czasie polilogarytmicznym. Nie są to jednak algorytmy typu PTAS. Polegają one z reguły na deterministycznym znajdowaniu rozwiązania frakcyjnego, a następnie na losowym zaokrąglaniu, tj. są to prawie wyłącznie algorytmy losowe. Algorytmy opisane w pracach [**h1**]-[**h6**] są deterministyczne, także działają w czasie polilogarytmicznym i dostarczają rozwiązań prawie dokładnych (typu PTAS/FPTAS). W porównaniu z metodą LP, ich wadą jest to, że są przeznaczone jedynie dla grafów, w których potrafimy obliczać klastry, np. dla grafów planarnych. Dodajmy, że są to algorytmy dla modelu LOCAL, a nie CONGEST, ponieważ prowadzenie obliczeń

[1] Polynomial Time Approximation Scheme; w modelu rozproszonym literka „P" oznacza w zasadzie Polylogarithmic, a nie Polynomial.

[2] Fully Polynomial Time Approximation Scheme; w pracach [**h1**]-[**h5**] współczynnik aproksymacji jest postaci $1 \pm \frac{1}{\log^{O(1)} n}$



Rysunek 1: Klastry w grafie służące do aproksymowania problemów grafowych.

w klastrach wymaga długich komunikatów. Kwestia czy można aproksymować (w sensie PTAS) problemy grafowe w modelu CONGEST, nawet ograniczając się do grafów planarnych, pozostaje otwarta.

W pracach grupy szwajcarskiej, np. w [8], znaleźć można także dolne oszacowania, $\Omega(\sqrt{\log n})$ i $\Omega(\log \Delta)$, na czas działania algorytmów rozproszonych znajdujących stałą lub logarytmiczną aproksymacje klasycznych problemów grafowych, w dowolnych grafach. Naszym wkładem w wiedzę o dolnych oszacowaniach jest przedstawione w [**h6**] nietrywialne dolne oszacowanie czasu potrzebnego do stałej aproksymacji MIS w cyklu.

5 Główne wyniki rozprawy.

Najprostsza metoda rozwiązywania problemów grafowych w modelu LOCAL polega na „zcentralizowaniu obliczeń", tj. wyznaczeniu lidera, który następnie zbierze opis całego grafu (czyli zbiór wszystkich krawędzi), oraz lokalnie rozwiąże zadany problem. Oczywiście nie stosujemy tej metody z uwagi na długi czas zbierania opisu całego grafu, równy jego średnicy. Wydaje się jednak, że idea „częściowej centralizacji obliczeń", czyli podział grafu na klastry o małej średnicy, patrz rysunek 1, oraz prowadzenie obliczeń opisaną wyżej metodą w klastrach, ma sens.

W omawianych niżej pracach [**h1**]-[**h6**] ogólny schemat rozproszonego algorytmu aproksymacyjnego, wykorzystującego klastry, jest następujący:

1. Znajdź w grafie klastry,

2. równolegle, w wszystkich klastrach, znajdź dokładne rozwiązanie problemu,

3. (jeśli to potrzebne) przystosuj rozwiązania w klastrach do całego grafu.

Punkt 2 jest oczywiście wykonalny, gdyż każdy klaster ma stałą średnicę oraz dopuszcza się dowolnie długie komunikaty. Zatem lider klastra może „zobaczyć" cały swój klaster i obliczyć jego dowolną funkcję. Oprócz małej średnicy wymaga się od klastrów pewnej dodatkowej własności gwarantującej, że punkt 3 nie będzie potrzebny lub że da się go wykonać. Własność ta zależy od konkretnego problemu co oznacza, że sam algorytm obliczający klastry nie wystarczy aby rozwiązać wszystkie problemy. Ta dodatkowa własność klastrów zazwyczaj sprowadza się do tego, że brzeg międzyklastrowy musi być w jakimś sensie „mały" w porównaniu do rozmiaru optymalnego rozwiązania problemu, przez co błędne decyzje algorytmu na brzegu klastrów nie będą miały wielkiego znaczenia dla jakości rozwiązania w całym grafie.

W pracy [**h1**] przedstawiliśmy sposób obliczania klastrów w drzewie, oraz pokazaliśmy jak aproksymować przy ich pomocy problemy MWM [3] i MWDS [4] w drzewach. Drugi algorytm w tej pracy oblicza klastry w grafie planarnym oraz znajduje rozwiązanie przybliżone problemu MWIS [5]. Warto zauważyć, że klastry drugiego algorytmu nie umożliwiają rozwiązania problemów MWM i MWDS w grafie planarnym, a rozwiązanie problemu MWIS jest możliwe tylko dzieki specjalnej cesze MWIS w grafach planarnych (związanej z powszechnie znanym faktem o 4-kolorowalności grafów planarnych).

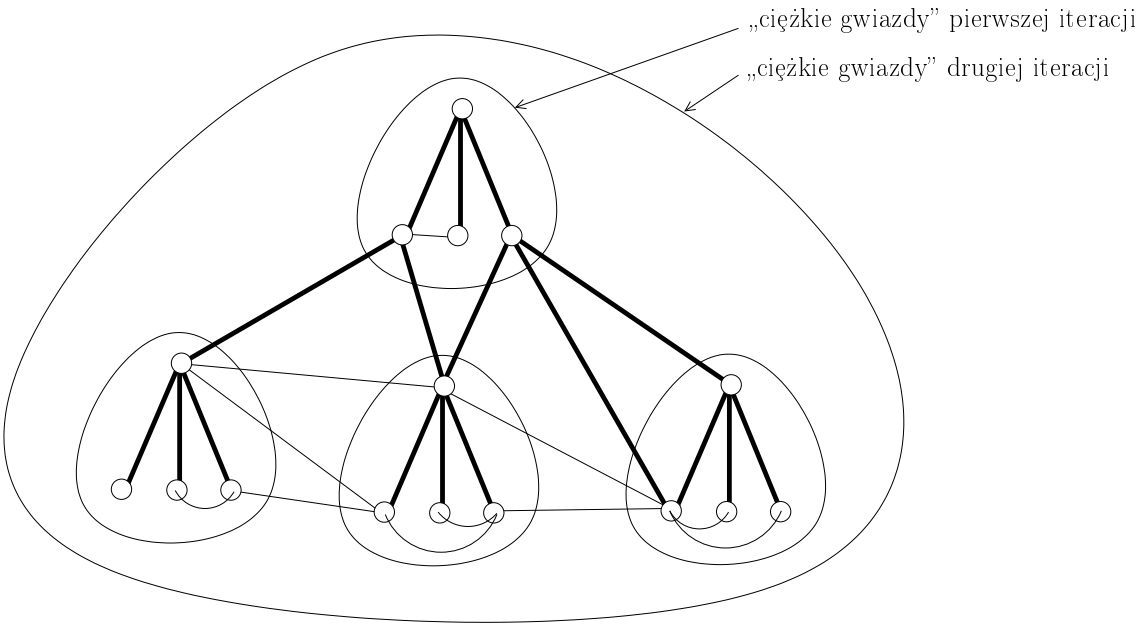
Sposób obliczania klastrów w drzewie pochodzi od następującej prostej obserwacji: jeśli w drzewie mamy q klastrów, to liczba *konektorów* (krawędzi łączących klastry) jest równa $q-1$. Zatem im mniej klastrów tym mniej konektorów i, co za tym idzie, mniejszy „brzeg międzyklastrowy". Liczbę klastrów można kontrolować przez wprowadzenie dolnego ograniczenia na ich rozmiar (jeśli klastry są duże to musi ich być mało). W przypadku nieważonym klastry w drzewie można zbudować przy pomocy dobrze znanej struktury zwanej „rulling set", patrz [10], która jest zbiorem równomiernie rozrzuconych wierzchołków w grafie. Jednak jeśli chcemy ich użyć do rozwiązywania ważonych problemów grafowych potrzebne jest nieco inne podejście; w omawianej pracy klastry mają własność, że dla każdego konektora o wadze ω łączącego dwa klastry X_1 i X_2, istnieją ścieżki w obu klastrach o długości $\Omega(\log n)$ złożone z krawędzi o wadze $\geq \omega/2$. Taka cecha klastrów oznacza, że waga „brzegu miedzyklastrowego" jest mała w stosunku do wagi optymalnego rozwiązania problemów MWM i MWDS w grafie wejściowym.

Klastry w grafie planarnym są budowane w inny sposób, poprzez technikę „ściskania ciężkich gwiazd", patrz rysunek 2. Polega ona na tym, że oblicza się w grafie rozłączne wierzchołkowo gwiazdy, ważące stałą frakcję wagi krawędziowej całego grafu i następnie

[3] Maximum Weighted Matching czyli ważone skojarzenie.

[4] Minimum Weighted Dominating Set czyli ważony minimalny zbiór dominujący wierzchołków.

[5] Maximum Weighted Independent Set czyli ważony maksymalny zbiór niezależny wierzchołków.



Rysunek 2: Budowanie klastrów w grafie planarnym.

się je „ściska" do pojedynczych wierzchołków. Tworzymy w ten sposób pewien wirtualny graf, który znów jest planarny. Powtarza się tę operację kilka razy. Wirtualne wierzchołki ostatniego grafu reprezentują klastry. Tak otrzymane klastry oprócz małej średnicy charakteryzują się więc małą wagą konektorów w stosunku do wagi krawędziowej całego grafu. Można je więc stosować do aproksymacji w przypadku problemów takich jak MWIS, których optymalne rozwiązanie waży dużo w porównaniu do wagi całego grafu, ale nie można stosować do problemów typu MDS/MM. Dodajmy, że w następne dwie prace stanowią próbę zastosowania tej techniki budowania klastrów właśnie do problemów MDS/MM...

W pracy [**h2**] próbujemy rozwiązać w grafach planarnych te problemy, z którymi potrafimy się uporać w drzewach, tj problemy MM i MDS (czyli Maximum Matching i Minimum Dominating Set, ale tym razem w wersji nieważonej). Okazuje się, że aproksymowanie problemu MM jest możliwe, o ile wcześniej przeprowadzi się specjalny preprocessing, który nie zmniejsza rozmiaru największego skojarzenia, ale zmniejsza liczbę wierzchołków grafu, tak, że największe skojarzenie jest duże w porównaniu do liczby wierzchołków. Co do problemu MDS, to praca przedstawia rozwiązanie częściowe, które wymaga aby graf komunikacyjny spełniał pewien dodatkowy warunek.

W pracy tej omówiono także dokładniej procedurę budowania klastrów w grafie planarnym, poprzez znajdowanie „ciężkich gwiazd" i ich „ściskanie". Lemat 4 stwierdza, że procedura budująca klastry gwarantuje, że mają one średnicę $O(\log^d n)$, gdzie $d := c \log 3 / \log \frac{1}{1-\frac{9}{10}\kappa} < 5.54c$, przy czym c jest parametrem procedury budującej klastry, a $\kappa = \frac{1}{st(G)}$, gdzie $st(G)$ to minimalna liczba lasów gwiazdowych, na które można rozbić

graf planarny. Jak wiadomo $st(G) = 5$ dla grafu planarnego G. Ponadto lemat stwierdza, że liczba konektorów jest ograniczona przez $O(|E(G)|/\log^c n)$, oraz, że czas działania procedury klastrujacej to $O(\log\log n \log^* n \log^{d+1} n)$.

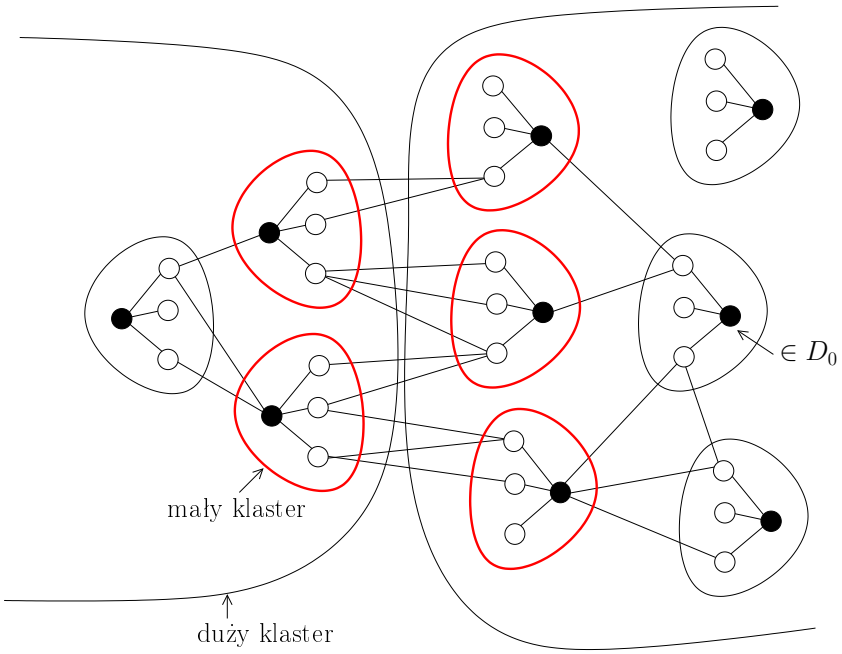
W pracy [**h3**] zauważamy, że algorytmy używające klastry w grafach planarnych działają także w szerszej klasie grafów, tzw. klasie *zamkniętej na minory*. Minor grafu G to tenże graf po „ściśnięciu" niektórych krawędzi i/lub usunięciu pewnych wierzchołków. Klasa zamknięta na minory to taki zbiór grafów[6], że jeśli weźmiemy dowolny jego element, to jego minor także będzie należał do tego zbioru. Klasa ta, oprócz grafów planarnych, zawiera wiele innych, np. grafy z ograniczonym parametrem treewidth. Wydaje się, że jest to największa klasa grafów, w której można obliczać klastry.

Główny wynik tej pracy to algorytm aproksymujący MDS, nie wymagający żadnych dodatkowych założeń co do grafu. Okazuje się, że aby to osiągnąć nie można używać standardowych klastrów, lecz muszą one być przystosowane do specyfiki problemu MDS (jest to nieco inne podejście niż w przypadku problemu MM, gdzie wystarczył preprocessing grafu i standardowe klastry). Oprócz problemu MDS udaje się rozwiązać jego odmianę MCDS[7], w którym graf indukowany przez zbiór dominujący jest spójny. Problem MCDS jest szczególnie ważny, gdyż posiada bezpośrednie praktyczne zastosownie: umożliwia zbudowanie „sieci szkieletowej" o możliwie małej liczbie wierzchołków; taka sieć upraszcza routing, gdyż musi się on odbywać wyłącznie w obrębie „szkieletu", a pozostałe węzły sieci maja maja bezpośrednie połączenia z węzłami „szkieletu".

Przypomnijmy tu definicję problemu MDS: zbiorem dominującym nazywamy zbiór wierzchołków D taki, że każdy wierzchołek albo należy do D albo ma sąsiada w D; MDS to zbiór dominujący najmniejszej mocy. Podstawowe niebezpieczeństwo związane z zastosowaniem klastrów do aproksmowania MDS polega na tym, że jest możliwa sytuacja, w której wierzchołki brzegowe danego klastra C wymagają dużej liczby dominatorów z wnętrza C, natomiast mogą być zdominowane przez małą liczbe wierzchołków spoza klastra C. Tak więc obliczanie MDS w klastrach prowadzi do znalezienia zbyt dużego zbioru wierzchołków. W pracy [**h3**] przedstawiamy zmodyfikowany sposób konstrukcji klastrów, który wyklucza taką sytuację, patrz rysunek 3. Sposób ten polega na dwupoziomowym budowaniu klastrów: najpierw buduje się „małe klastry" o specjalnej własności, potem się je ściska do pojedynczych wierzchołków, a następnie w powstałym wirtualnym grafie po raz drugi oblicza się klastry, które stają się „dużymi klastrami" po powrocie do fizycznego grafu. Małe klastry są budowane na bazie zbioru D_0, który jest stałą aproksymacją problemu MDS. Zbiór D_0 trzeba obliczyć na początku, osobnym algorytmem. Liczba małych klastrów brzegowych jest mała w porównaniu z $|D_0|$, co oznacza, że można zdominować

[6] Różny od zbioru wszystkich grafów!

[7] Minimum Connected Dominating Set



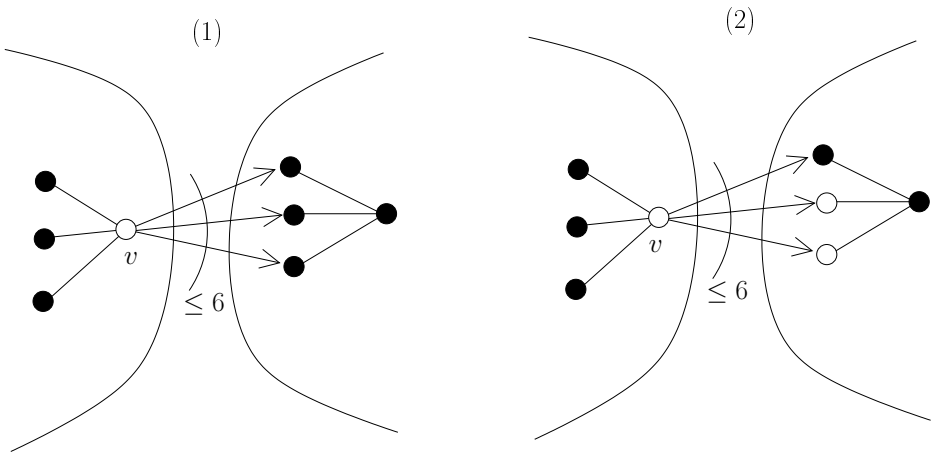
Rysunek 3: Specjalna klastry dla problemu MDS.

od środka brzeg dużych klastrów przy pomocy małej liczby wierzchołków (małej w porównaniu do rozmiaru optymalnego rozwiązania MDS). Zatem niebezpieczeństwo, o którym mowa wyżej nie istnieje...

W pracy [**h4**] podajemy algorytmy aproksymujące dowolnie dokładnie ważone problemy MWM i MWDS w klasie grafów zamkniętej na minory.

Okazuje się, że ważone wersje problemów MM i MDS wymagają innego typu klastrów, w pewnym sensie „silniejszych" niż te używane w poprzednich pracach, a także trudniejszych do obliczenia. Są to tzw. *klastry wierzchołkowe*, zdefiniowane dla grafu z wagami na wierzchołkach, charakteryzujące się tym, że waga wierzchołków brzegowych klastrów jest mała w porównaniu z wagą wierzchołków całego grafu (w poprzednio używanych klastrach, które można nazwać *krawędziowymi*, waga krawędzi międzyklastrowych była mała w porównaniu z wagą krawędzi całego grafu). Zauważmy, że w przypadku problemu MWM nie możemy zastosować strategii używanej w przypadku nieważonego problemu MM, gdyż nie jest znany preprocessing grafu, który gwarantuje, że rozmiar optymalnego rozwiązania MWM stanie się zbliżony do wagi krawędziowej całego grafu.

Procedura budowania klastrów wierzchołkowych polega na tworzeniu „skupisk" wierzchołków, które ważą stałą frakcję wagi wierzchołkowej całego grafu oraz powodowaniu, że skupiska te będą odpowiednio odseparowane (będą w odległości ≥ 2). Skupiska te stają się wnętrzami klastrów, a przez to, że wnętrza maja dużą wagę, brzegi (wierzchołkowe) klastrów muszą mieć małą wagę. Budowanie skupisk zaczyna się od zbudowania takich, które są odseparowanych jedynie o 1, następnie usuwa się pewne wierzchołki, tak aby zachować dużą wagę skupisk, a równocześnie zwiększyć ich separację. Aby taka oper-



Rysunek 4: Upraszczanie brzegu podczas budowy skupisk (dla klastrów wierzchołkowych).

acja była wykonalna „upraszcza się" brzeg miedzy skupiskami. Jeden z takich kroków, upraszczających brzeg, pokazano na rysunku 4. Na rysunku tym skupiska składają się z czarnych wierzchołków. Wierzchołek v jest odpowiedzialny za to, że odległość między skupiskami wynosi 1 zamiast 2. Pewne wierzchołki zostaja pomalowane na biało w taki sposób, że wprawdzie v nadal jest odpwiedzialny za zbyt mała separację obu skupisk, ale sytuacja staje się prostsza...

Klastry wierzchołkowe z poprzednio omawianej pracy znajdują dodatkowe zastosowanie w pracy [**h5**] do „pakowania" małych podgrafów w grafie planarnym. Problem pakowania w grafie G polega na wybraniu z pewnego zbioru podgrafów dopuszczalnych, podzbioru rozłącznych wierzchołkowo podgrafów, o możliwie dużej mocy. Od podgrafów wymaga się jedynie aby składały się ze stałej liczby wierzchołkow, a ich „dopuszczalność" może byc zdefiniowana dowolnie. Tak więc mogą to być np. wszystkie krótkie ścieżki łączące pary wierzchołków (s_i, t_i), $i = 1, \ldots, k$, co oznacza, że szukamy jak największej liczby rozłącznych wierzchołkowo ścieżek łączących te pary.

Idea algorytmu jest następująca: na początku przy pomocy klastrów krawędziowych budujemy w grafie specjalne „małe klastry" oraz zbiór wierzchołków Z, który jest pokryciem wierzchołkowym konektorów małych klastrów. Dodatkowo moc Z jest niewiele większa od mocy optymalnego pakowania dopuszczalnych podgrafów. Analiza algorytmu konstruującego małe klastry i zbiór Z wymaga reprezentacji graficznej grafu planarnego, dlatego cały algorytm jest przeznaczony dla grafów planarnych, a nie dla klasy zamkniętej na minory. Następnie ściskamy małe klastry do pojedynczych wierzchołków i definiujemy ich wagę jako liczbę wierzchołków Z wewnątrz ściśniętego klastra. W otrzymanym wirtualnym grafie obliczamy klastry wierzchołkowe, używając procedury z poprzedniej pracy i otrzymujemy „duże klastry" w grafie wejściowym, które mają bardzo mało wierzchołków zbioru Z na brzegach. Innymi słowy pokrycie wierzchołkowe konektorów dużych klastrów jest małe. Teraz pakujemy optymalnie dopuszczalne podgrafy w dużych klastrach.

Postępując tak pomijamy pewne podgrafy dopuszczalne znajdujące się na miedzyklastrowym brzegu. Niewątpliwie podgrafy te muszą zawierać konektory tworzące skojarzenie. Z drugiej strony wiemy, że pokrycie wierzchłkowe konektorów dużych klastrów jest bardzo małe w porównaniu do rozmiaru optymalnego pakowania. Zatem możemy stwierdzić, że nasze rozwiązanie jest dobrej jakości, powołując się na znany fakt, że moc pokrycia wierzchołkowego jest górnym ograniczeniem mocy skojarzenia.

W pracy [**h6**] podajemy m.in. ulepszoną wersję algorytmu obliczającego klastry krawędziowe w grafach planarnych, który tym razem działa w znacznie krótszym czasie $O(\log^* n)$. Pozwala to na obliczanie $(1 \pm \epsilon)$-aproksymacji problemów MWIS, MDS i MM, w czasie $O(\log^* n)$ dla stałego ϵ. W poprzednich pracach klastry były obliczane w czasie $O(\log n \log^* n)$ lub dłuższym. Omawiana praca jest jedną z pierwszych, w których wszystkie algorytmy działają w czasie sublogarytmicznym (wcześniej często zadowalano się czasem polilogarytmicznym i tak było też w przypadku prac [**h1**]-[**h5**]). Podobną tendencję do rozpatrywania algorytmów synchronicznych działających w bardzo krótkim czasie obserwuje się obecnie często, np. grupa finska, Suomela i inni, [17], rozpatrują algorytmy ściśle lokalne, czyli działające w stałym czasie. Krótki czas działania algorytmu można do pewnego stopnia powiązać z odpornością na błędy lub z tzw. *samostabilizacją*, gdyż jest możliwe cykliczne powtarzanie takiego algorytmu dzięki czemu jeśli w sieci pojawi się zmiana lub uszkodzenie to algorytm szybko odtworzy prawidłowe rozwiązanie problemu. Zauważmy też, że szybka procedura obliczania klastrów dotyczy wyłącznie klastrów krawędziowych, a nie wierzchołkowych, zatem nie oznacza, że można szybko aproksymować problemy typu MWM czy MWDS lub pakowanie małych podgrafów (pozostaje kwestią otwartą czy jest w ogóle możliwe rozwiązywanie tych problemów w czasie sublogartmicznym).

Praca [**h6**] zawiera dodatkowo dowód, że nie da się w stałym czasie znaleźć stałej aproksymacji problemu MIS w cyklu, wykorzystujący twierdzenie Ramseya. Zawiera także algorytm losowy, który w stałej liczbie rund i z dużym prawdopodobieństwem znajduje $(1 - \delta)$-aproksymację problemu MIS w grafie planarnym, dla dowolnego stałego δ.

W pracach [**h7**], [**h8**] i [**h9**] skupiamy się na aproksymacji rozproszonej w grafach, w których nie można budować klastrów. Wydaje się, że najprostszą klasą grafów o tej własności są grafy o *stałej drzewiastości*. Są to takie grafy, które można przedstawić jako sumę stałej liczby lasów. Klasa ta zawiera w szczególności klasę grafów zamkniętą na minory.

W pracy [**h7**] opisaliśmy algorytm znajdujący $(1 - \epsilon)$-aproksymację problemu MM w grafach o stałej drzewiastości, działający w czasie $O(\log^* n)$. Opieramy się w nim na dwóch faktach:

1. W grafie o stałym stopniu można obliczać $(1-\epsilon)$-aproksymacje problemu MM w czasie $O(\log^* n)$, poprzez obliczanie MIS w wirtualnym grafie, którego wierzchołkami są ścieżki rozszerzające, używane standardowo do aproksymowania skojarzeń,

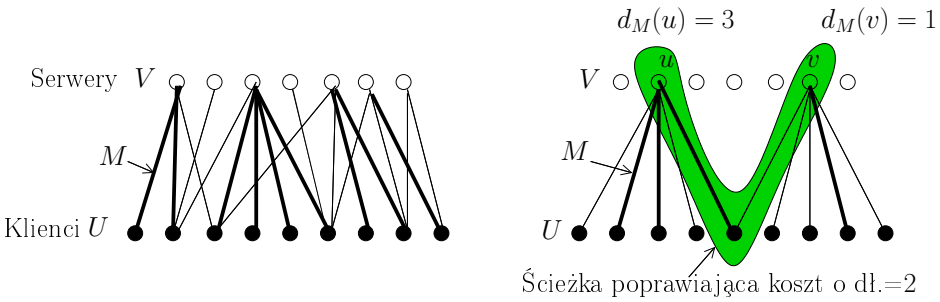
2. w grafie dwudzielnym (X, Y, E), gdzie X i Y to zbiory wierzchołków, można obliczyć $(1-\epsilon)$-aproksymację MM w stałym czasie, o ile tylko maksymalny stopień wierzchołków na jednym ze zbiorów X lub Y jest stały, także posługując się ścieżkami rozszerzającymi.

Zauważmy, że oba wyżej wymienione rodzaje grafów to szczególne przypadki grafów o stałej drzewiastości. W pracy [**h7**] udało nam się uzyskać algorytm dla ogólnego grafu o stałej drzewiastości. Praca ta udowodniła więc, że jest możliwe szybkie i dokładne aproksymowanie skojarzeń, także w grafach, w których nie można obliczać klastrów. Nie da się tego powiedzieć o innych problemach grafowych, takich jak MIS czy MDS, stąd idea aby szerzej „eksploatować" problemy przypominające skojarzenia.

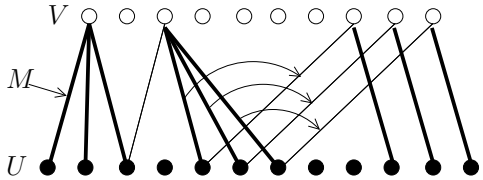
W pracach [**h8**] i [**h9**] opisaliśmy algorytmy znajdujące stałą aproksymację *semiskojarzeń* w grafach dwudzielnych, spełniających różne założenia.

Niech $G = (U, V, E)$ będzie grafem dwudzielnym, gdzie U i V to zbiory wierzchołków, a E jest zbiorem krawędzi. Oznaczmy przez $d_Q(v)$ stopień wierzchołka v w podgrafie Q grafu G, a przez $\Delta(V)$ oznaczmy maksmalny stopień na zbiorze V. Semiskojarzeniem w grafie G nazywamy podgraf M taki, że $\forall_{u \in U} d_M(u) = 1$, przy czym koszt rozwiązania (który minimalizujemy) definiujemy jako $\sum_{v \in V} d_M^2(v)$ lub podobnie. Problem semiskojarzeń występuje w literaturze pod różnymi nazwami, w szczególności jako „load balancing". Posiada on następującą praktyczną motywację: wierzchołki zbioru U możmy uważać za (software-owych) „klientów" podłączonych do „serwerów" ze zbioru V. Każdy klient potrzebuje dokładnie jednego serwera, przy czym obciążenie serwerów (czyli liczba obsługiwanych przez nich klientów, $d_M(v)$) powinno być w miarę możliwości równe. Dlatego właśnie minimalizujemy sumę kwadratów stopni $d_M(v)$. Istnieje obszerna litaratura na temat semiskojarzeń, [21], [20], [22], jednak omawia ona z reguły algorytmy sekwencyjne lub rozproszone losowe. Główna technika aproksymowania semiskojarzeń polega na znajdowaniu tzw. *ścieżek poprawiających koszt*, patrz rysunek 5. Są to ścieżki parzystej długości, o końcach u i v w zbiorze V, składające się na przemian z krawędzi z M i z $E \setminus M$, spełniające dodatkowo warunek $d_M(u) > d_M(v) + 1$. Ścieżki takie następnie się „rotuje" aby zmniejszyć różnicę stopni na u i v.

Podstawowym składnikiem pracy [**h9**] jest algorytm rozproszony przeznaczony dla grafów spełniających warunek: $\Delta(V) \leq \text{const}$. Algorytm ten w stałym czasie znajduje 2-aproksymację optymalnego semiskojarzenia eliminując wszystkie ścieżki poprawiające koszt o długości 2. Robi to znajdując duże zbiory takich ścieżek i rotując ich zawartość.



Rysunek 5: Semiskojarzenie i ścieżka poprawiająca koszt dł.=2.



Rysunek 6: Nieudane rotowanie ścieżek poprawiających koszt dł.=2.

Okazuje się, że jest to czynność nietrywialnym, co widać na rysunku 6. Gdyby zrotować wszystkie ścieżki poprawiające koszt długości 2 widoczne na rysunku, to powstałyby nowe ścieżki poprawiające koszt!

W pracy opisano także algorytm typu „greedy" wykorzystujący tzw. q-skojarzenia maximal. Algorytm ten znajduje stałą aproksymacje optymalnego semiskojarzenia w grafach spełniających pewne dodatkowe warunki, a jego analiza wykorzystuje analizę algorytmu greedy dla problemu MSSC (Minimum Sum Set Cover), opisanego w pracy [20]. Podgraf Q jest q-skojarzeniem jeśli $\forall_{u \in U} d_Q(u) \in \{0, 1\}$ oraz $\forall_{v \in V} d_Q(v) \leq q$. Okazuje się, że stałą aproksymację semiskojarzeń można zbudować z q-skojarzeń maximal, o ile w grafie wejściowym istnieje optymalne semiskojarzenie M^* o własności $\forall_{v \in V} d_{M^*}(v) \geq q$. W pracy [**h9**] oraz w jej wersji czasopismowej (jeszcze nie zaakceptowanej przez czasopismo JSCC, Journal of Computer and System Sciences) podajemy przykłady grafów, w których opisane wyżej podejście gwarantuje szybkie znalezienie stałej aproksymacji semiskojarzania. Wydaje się także, że to podejscie pozwoli w przyszłości rozwiązać problem w ogólnym grafie dwudzielnym, w czasie polilogarytmicznym.

6 Pozostały dorobek naukowy.

W pracach [p1], [p2], [p3], [p4] używamy narzędzia zwanego *spannerem*, które powstało w czasie prac nad algorytmem obliczającym MM czyli Maximal Matching, w dowolnych grafach [25], stanowiącym część mojej pracy doktorskiej. Niech $G = (A, B, E)$ będzie tzw. D-blokiem czyli grafem dwudzielnym, ze zbiorami wierzchołków A i B oraz zbiorem krawędzi E, spełniającym warunek: $\forall_{v \in A} : d(v) \in [D/2, D]$. Spannerem nazywamy

podgraf S grafu G taki, że:

1. S zawiera $\frac{1}{2}$ wierzchołków zbioru A,
2. $\forall_{v \in A} : d_S(v) = 1$,
3. $\forall_{v \in B} : d_S(v) < \frac{O(1)}{D} d(v) + 1$,

przy czym $d(v)$ oznacza stopień w G, a $d_S(v)$ oznacza stopień w podgrafie S. W modelu rozproszonym można obliczyć spanner w czasie $O(\log^3 n)$, gdzie $n = |A| + |B|$.

Poprzednio spanner był używany do obliczania w D-bloku dużych skojarzeń. Okazuje się, że ma on także inne zastosowania: W pracy [p1] używamy rodziny spannerów do kolorowania krawędziowego grafu przy pomocy $O(\Delta \log n)$-kolorów. Kolorowanie rozproszone jest w dziedzinie obliczeń rozproszonych problemem równie istotnym jak MIS. W ostatnich latach widac na tym polu dalszy postęp, w postaci prac Elkina i Barenboima, [23], [24].

Spanner może też być użyty do pakowania ścieżek rozszerzających w tzw. grafie warstwowym, co pozwala aproksymować skojarzenia. W pracy [p2] podaliśmy algorytm znajdujący $\frac{2}{3}$-aproksymację skojrzenia, działający w czasie $O(\log^6 n)$. W tym algorytmie spanner jest używany do pakowania ścieżek rozszerzających długości 3, a cały algorytm działa dla dowolnego grafu. W pracy [p3] opisaliśmy algorytm znajdujący $(1-\epsilon)$-aproksymację skojarzeń w grafie dwudzielnym, działający w czasie $O(\log^{O(1/\epsilon)} n)$. Algorytm ten stosuje pakowanie ścieżek rozszerzających o długości $2k+1$, znów przy pomocy spannerów. W pracy [p4] poprawiliśmy czas działania algorytmu z [p2] do $O(\log^4 n)$, poprzez wprowadzenie większej równoległości obliczeń w tzw. (D_1, D_2)-blokach. A zatem dysponujemy algorytmem obliczającym $\frac{2}{3}$-aproksymację skojarzeń, działającym w takim samym czasie jak algorytm znajdujący $\frac{1}{2}$-aproksymacje (i jednocześnie skojarzenie typu Maximal).

Jeśli chodzi o aproksymację przy pomocy klastrów, to oprócz klasy grafów zamkniętej na minory, rozważaliśmy także klasę grafów dyskowych, czyli UDG (Unit Disk Graph). Są to grafy w których każdy wierzchołek jest reprezentowany przez okrąg o promieniu 1 na płaszczyźnie, a krawędź między wierzchołkami istnieje jeśli ich okręgi sie przecinają (czyli wierzchołki są w odległości ≤ 2 na płaszczyźnie). Ten typ grafów jest motywowany sieciami radiowymi, w których komunikować się mogą wyłącznie wierzchołki leżące blisko siebie. Problemy grafowe w UDG mają bogatą literaturę, patrz [18], [19]. Sieci te często występują także pod nazwą „adhoc networks" lub „sensor networks".

Okazuje się, że w tego typu grafach także można budować klastry w sposób rozproszony. Wykorzystuje się tu cechę grafów UDG o nazwie „bounded growth", polegającą na tym, że w kuli o promieniu k jest co najwyżej $O(k^2)$ niezależnych wierzchołków. Tak więc budując kule o coraz większym promieniu skonczymy na kuli o „małym brzegu" (a dokładniej z małym MIS-em na brzegu), która może być używana jako klaster. W pracy

[p5] przedstawiliśmy kilka algorytmów aproksymacyjnych, dla klasycznych problemów grafowych, używających klastrów budowanych w wyżej opisany sposób. Natomiast w pracy [p6] przedstawiliśmy algorytm pakujący małe podgrafy w grafie UDG, podobny do algorytmu z [**h5**] dla grafów planarnych.

Literatura

[1] B. Awerbuch: Complexity of network synchronization Journal of the ACM, Volume 32 Issue 4, Oct. 1985, pp. 804 - 823.

[2] D. S. Hirschberg, J. B. Sinclair: Decentralized extrema-finding in circular configurations of processors, Communications of the ACM, Volume 23 Issue 11, Nov. 1980, pp 627 - 628.

[3] Y. Afek, E. Gafni: Time and message bounds for election in synchronous and asynchronous complete networks, PODC '85, Proceedings of the 4th annual ACM symposium on Principles Of Distributed Computing, pp. 186-195.

[4] R. G. Gallager, P. A. Humblet, and P. M. Spira: A distributed algorithm for minimum-weight spanning trees, ACM Transactions on Programming Languages and Systems, vol. 5, no. 1, January 1983, pp. 66-77.

[5] P. Marshall, R. Shostak, L. Lamport: Reaching Agreement in the Presence of Faults, Journal of the Association for Computing Machinery 27 (2), April 1980.

[6] F. Kuhn, R. Wattenhofer: Constant-time distributed dominating set approximation, PODC '03, Proceedings of the 23 annual symposium on Principles Of Distributed Computing, pp. 25-32.

[7] F. Kuhn, R. Wattenhofer: Distributed combinatorial optimization, raport techniczny, http://e-collection.library.ethz.ch/eserv/eth:4623/eth-4623-01.pdf

[8] F. Kuhn, T. Moscibroda, R. Wattenhofer. What Cannot Be Computed Locally! 23rd ACM Symposium on the Principles of Distributed Computing (PODC), St. John's, Newfoundland, Canada, July 2004.

[9] R. Cole, U. Vishkin: Deterministic coin tossing and accelerating cascades: micro and macro techniques for designing parallel algorithms, STOC '86, Proceedings of the eighteenth annual ACM symposium on Theory of computing, pp. 206 - 219.

[10] B. Awerbuch, A.V. Goldberg, M. Luby, and S. Plotkin: Network decomposition and locality in distributed computing, Proceedings of the 30th Symposium on Foundations of Computer ScienceFOCS 1989, (1989), 364–369.

[11] N. Linial: Locality in distributed graph algorithms, SIAM Journal on Computing, 21(1), (1992), pp. 193–201.

[12] N. Linial and M. Saks: Low diameter graph decompositions, Combinatorica, 13(1993), pp. 441-454.

[13] A. Panconesi, A. Srinivasan: On the Complexity of Distributed Network Decomposition, Journal of Algorithms, Volume 20, Issue 2, March 1996, pp. 356-374

[14] Z. Lotker, B. Patt-Shamir, A. Rosén: Distributed Approximate Matching, SIAM J. Comput. 39(2), (2009), pp. 445–460.

[15] Ch. P. Low: An approximation algorithm for the load-balanced semi-matching problem in weighted bipartite graphs, Information Processing Letters, v.100 n.4, (2006), pp. 154–161.

[16] D. Peleg: Distributed Algorithms, A Locality-Sensitive Approach; SIAM Press, (2000).

[17] J. Suomela: Survey of Local Algorithms, (manuscript), available online http://www.cs.helsinki.fi/u/josuomel/doc/local-survey.pdf

[18] F. Kuhn, T. Moscibroda, R. Wattenhofer: On the locality of bounded growth, PODC '05, Proceedings of the 24 annual ACM symposium on Principles of distributed computing, pp. 60-68 .

[19] J. Schneider, R. Wattenhofer: A log-star distributed maximal independent set algorithm for growth-bounded graphs, PODC '08, Proceedings of the 27 ACM symposium on Principles of distributed computing, pp. 35-44.

[20] U. Feige, L. Lovasz, P. Tetali: Approximating Min Sum Set Cover, Algorithmica 40(4), (2004) pp. 219–234.

[21] N. J. A. Harvey, R. E. Ladner, L. Lovasz, T. Tamir: Semi-matchings for bipartite graphs and load balancing, J. Algorithms 59 (1), (2006), pp. 53–78.

[22] F. Galčík, J. Katrenič, G. Semanišin: On computing an optimal semi-matching, Proceedings of the 37th international conference on Graph-Theoretic Concepts in Computer Science, (2011), pp. 250–261.

[23] L. Barenboim and M. Elkin: Distributed Deterministic Edge Coloring using Bounded Neighborhood Independence, To appear in Distributed Computing, Special Issue for PODC'11 conference.

[24] L. Barenboim and M. Elkin: Distributed Deterministic Vertex Coloring in Polylogarithmic Time, In Journal of ACM, Vol. 58, No. 5, 23, 2011.

Moje prace przed doktoratem (chronologicznie) .....

[25] M. Hańćkowiak, M. Karonski, A. Panconesi: On the distributed complexity of computing maximal matchings, Proceedings of SODA 98, the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, pp. 219-225.

[26] M. Karonski, A. Panconesi, M. Hańćkowiak: A faster distributed algorithm for computing maximal matching deterministically, PODC1999, Proceedings of the Eighteenth Annual ACM Symposium on Principles of Distributed Computing, Atlanta, Georgia, USA, May 3-6, 1999, pp. 219-228.

[27] M. Karonski, A. Panconesi, M. Hańćkowiak: On the distributed complexity of computing maximal matchings, SIAM J. Discrete Mathematics, Vol. 15, No. 1, 2001, pp. 41-57.

Moje prace po doktoracie (chronologicznie) .....

[p1] A. Czygrinow, M. Karonski, M. Hańćkowiak: Distributed $O(Deltalogn)$-Edge-Coloring Algorithm, ESA'2001, Algorithms, 9th Annual European Symposium, Aarhus, Denmark, August 28-31, 2001, pp. 345-355.

[p2] A. Czygrinow, E. Szymanska, M. Hańćkowiak: Distributed algorithm for approximating the maximum matching, Discrete Applied Mathematics 143(1-3), 2004, pp. 62-71.

[p3] A. Czygrinow, M. Hańćkowiak: Distributed algorithm for better approximation of the maximum matching, COCOON 2003, Computing and Combinatorics, 9th Annual International Conference, Big Sky, MT, USA, July 25-28, 2003, pp. 242-251.

[p4] A. Czygrinow, M. Hańćkowiak, E. Szymanska: A Fast Distributed Algorithm for Approximating the Maximum Matching, ESA 2004, Algorithms, 12th Annual European Symposium, Bergen, Norway, September 14-17, 2004, pp. 252-263.

[**h1**] A. Czygrinow, M. Hańćkowiak: Distributed algorithms for weighted problems in sparse graphs, J. Discrete Algorithms 4(4), 2006, pp. 588-607

[**h2**] A. Czygrinow, M. Hańćkowiak, E. Szymanska: Distributed Approximation Algorithms for Planar Graphs, CIAC 2006, Algorithms and Complexity, 6th Italian Conference, Rome, Italy, May 29-31, 2006, pp. 296-307.

[**h3**] A. Czygrinow, M. Hańćkowiak: Distributed Almost Exact Approximations for Minor-Closed Families, ESA 2006, Algorithms, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, pp. 244-255.

[p5] A. Czygrinow, M. Hańćkowiak: Distributed Approximation Algorithms in Unit-Disk Graphs, DISC 2006, Distributed Computing, 20th International Symposium, Stockholm, Sweden, September 18-20, 2006, pp. 385-398.

[**h4**] A. Czygrinow, M. Hańćkowiak: Distributed Approximation Algorithms for Weighted Problems in Minor-Closed Families, COCOON 2007, Computing and Combinatorics, 13th Annual International Conference, Banff, Canada, July 16-19, 2007, pp. 515-525.

[p6] A. Czygrinow, M. Hańćkowiak: Distributed Approximations for Packing in Unit-Disk Graphs, DISC 2007, Distributed Computing, 21st International Symposium, DISC 2007, Lemesos, Cyprus, September 24-26, 2007, pp. 152-164.

[**h5**] A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Distributed packing in planar graphs, SPAA 2008, Proceedings of the 20th Annual ACM Symposium on Parallelism in Algorithms and Architectures, Munich, Germany, June 14-16, 2008, pp. 55-61.

[**h6**] A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Fast Distributed Approximations in Planar Graphs, DISC 2008, Distributed Computing, 22nd International Symposium, Arcachon, France, September 22-24, 2008, pp. 78-92.

[**h7**] E. Szymanska, A. Czygrinow, M. Hańćkowiak: Fast Distributed Approximation Algorithm for the Maximum Matching Problem in Bounded Arboricity Graphs, ISAAC 2009, Algorithms and Computation, 20th International Symposium, Honolulu, Hawaii, USA, December 16-18, 2009, pp. 668-678.

[**h8**] E. Szymanska, K. Krzywdzinski, A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Distributed Approximation Algorithm for the Semi-Matching Problem, DISC 2011 (Brief Announcements only !), Distributed Computing, 25th International Symposium, Rome, Italy, September 20-22, 2011, pp. 200-201.

[**h9**] E. Szymanska, A. Czygrinow, M. Hańćkowiak, W. Wawrzyniak: Distributed 2-approximation Algorithm for the Semi-Matching Problem, DISC 2012, Distributed Computing, 26th International Symposium, Salvador, Brazil, October 16-18, 2012, pp. 210-222.