Struktury danych i algorytmy dynamiczne dla grafów planarnych

Autoreferat rozprawy doktorskiej

Adam Karczmarz

Instytut Informatyki, Uniwersytet Warszawski

1 Wprowadzenie

Dla najbardziej podstawowych problemów grafowych, takich jak znajdowanie najkrótszych ścieżek bądź obliczanie najliczniejszych skojarzeń, wciąż nie są znane algorytmy, o których umiemy teoretycznie udowodnić, że zadziałają wystarczająco szybko dla każdego grafu o liczbie krawędzi rzędu dziesiątek lub setek milionów (a więc wielkości spotykanej w praktyce). Na przykład, jeśli skupimy się na tzw. algorytmach silnie wielomianowych (zob. np. [65]), nadal nie wiemy, jak obliczać najkrótsze ścieżki w grafach rzadkich o wagach krawędzi będących liczbami rzeczywistymi, istotnie szybciej niż w czasie kwadratowym, używając klasycznego, choć poniekąd oczywistego, algorytmu Bellmana-Forda.

Jednym ze sposobów na poradzenie sobie z tym problemem jest rozważanie mniej ogólnych modeli obliczeń dla algorytmów grafowych. Jeśli, przykładowo, ograniczymy się do przypadku, gdy wagi krawędzi grafu wejściowego są całkowitoliczbowe, jesteśmy w stanie uzyskać algorytmy o wiele szybsze niż metoda Bellmana-Forda [14, 31]. Choć algorytmy te stanowią głębokie wyniki, ich pesymistyczna złożoność czasowa jest wciąż daleka od linowej. Warto dodać, że nie są znane dotychczas żadne nietrywialne (tj. ponadliniowe) ograniczenia dolne na złożoność obliczeniową problemu znajdowania najkrótszych ścieżek w przypadku, gdy dopuszczamy ujemne wagi.

Innym możliwym podejściem jest projektowanie algorytmów przeznaczonych specjalnie dla pewnych klas grafów spotykanych w praktyce. Grafy planarne są jedną z najważniejszych i najlepiej zbadanych klas tego typu. Wiele spośród sieci występujących w otaczającym nas świecie może być narysowanych na płaszczyźnie bez przecięć lub też z małą ich ilością. Wśród przykładów znajdziemy niezbyt skomplikowane sieci drogowe, czy też grafy spotykane przy projektowaniu układów scalonych wielkiej skali integracji. Bardziej skomplikowane sieci drogowe, choć już dalekie od bycia planarnymi, dzielą z grafami planarnymi pewne ważne i użyteczne własności, takie jak istnienie małych separatorów [20]. Szczególne przypadki grafów planarnych – kraty – pojawiają się często w dziedzinie przetwarzania obrazów (np. [7]).

Istotnie, jeśli ograniczymy naszą uwagę do grafów planarnych, wiele spośród klasycznych wielomianowych problemów grafowych, takich jak obliczanie najkrótszych ścieżek [35, 58] i maksymalnych przepływów [4, 5, 21], da się rozwiązać w czasie prawie liniowym lub nawet optymalnie. Bogata struktura kombinatoryczna grafów planarnych często pozwala na pokonywanie przeszkód napotykanych w odpowiednich problemach dla grafów ogólnych za pomocą technik pochodzących z geometrii obliczeniowej (np. [27]) lub przy użyciu zaawansowanych struktur danych, takich jak dynamiczne drzewa [4, 10, 21, 66].

W tej rozprawie skupiamy się właśnie na strukturodanowym aspekcie algorytmiki grafów planarnych. Mówiąc ściślej, zamiast koncentrować się na konkretnych problemach grafowych,

rozważamy bardziej abstrakcyjne, „niskopoziomowe" problemy. Efektywne rozwiązania problemów tego typu mogą być użyte jako „czarne skrzynki" przy projektowaniu algorytmów dla konkretnych problemów na grafach planarnych. Takie podejście pozwala nam uzyskać szybsze algorytmy dla wielu problemów jednocześnie, bez zagłębiania się w szczegóły znanych algorytmów dla tychże problemów.

Badamy także *algorytmy dynamiczne* dla grafów planarnych, tj. algorytmy, które utrzymują pewien zbiór informacji o dynamicznie zmieniającym się grafie (przykładowo, informację czy graf jest spójny) o wiele efektywniej niż poprzez przeliczanie tych informacji od nowa po każdej drobnej zmianie grafu. Rozważamy model *krawędziowy*, w którym graf wejściowy może być zmieniany tylko poprzez wstawienia lub usunięcia pojedynczych krawędzi. Algorytm grafowy nazywamy *w pełni dynamicznym*, jeśli obsługuje zarówno wstawienia jak i usunięcia krawędzi. W przypadku gdy obsługiwane jest tylko dodawanie krawędzi (model *inkrementalny*) bądź tylko usuwanie krawędzi (model *dekrementalny*), algorytm nazywamy *częściowo dynamicznym*.

Przy projektowaniu dynamicznych algorytmów grafowych interesuje nas *czas zmiany*, tj. czas potrzebny algorytmowi na dostosowanie się do elementarnej zmiany grafu, i *czas zapytania*, czyli czas potrzebny algorytmowi na przeliczenie żądanej części utrzymywanej informacji. Czasami, szczególnie w modelach częściowo dynamicznych, wygodniej jest mierzyć *całkowity czas zmian*, czyli sumaryczny czas użyty przez algorytm do przetworzenia dowolnej sekwencji zmian. Dla pewnych problemów dynamicznych warto skupić się na jeszcze bardziej ograniczonym *jawnym* modelu, gdzie po każdej zmianie cała utrzymywana informacja o grafie musi być jawnie zaktualizowana. W takim modelu algorytm obsługujący zapytanie jest zawsze trywialny i polega na odczytaniu żądanej komórki pamięci, a więc interesuje nas wtedy tylko czas zmiany.

Zauważmy, że granica pomiędzy dynamicznymi algorytmami grafowymi i grafowymi strukturami danych nie jest jasno określona. Algorytmy dynamiczne są bowiem często używane w roli „czarnych skrzynek" do uzyskania efektywnych algorytmów *statycznych* (np. [26]). Przykładowo, problem *inkrementalnej spójności*, w którym potrzebujemy odpowiadać na zapytania o istnienie ścieżki pomiędzy danymi dwoma wierzchołkami grafu nieskierowanego, podczas gdy do grafu są dodawane nowe krawędzie, jest tak naprawdę równoważny problemowi *struktury danych dla zbiorów rozłącznych* (por. np. [15]), zwanym także problemem *union-find*.

W tej rozprawie koncentrujemy się głównie na modelu dekrementalnym. Pokazujemy efektywne dekrementalne algorytmy na nieważonych grafach planarnych dla pewnych podstawowych problemów związanych z osiągalnością i spójnością. Omawiamy także zastosowania uzyskanych algorytmów dynamicznych w algorytmach statycznych, potwierdzając w ten sposób ponownie strukturodanowy charakter tych wyników.

W dalszej części tego autoreferatu oznaczamy przez $G = (V, E)$ wejściowy graf planarny. Niech n oznacza liczbę wierzchołków grafu G. Dla uproszczenia zakładamy, że G jest grafem prostym, tzn. grafem bez pętli i krawędzi wielokrotnych. Wtedy, na mocy planarności, G ma $O(n)$ krawędzi. Gdy mówimy o grafach ogólnych, oznaczamy przez m liczbę krawędzi grafu.

2 Ściąganie grafu planarnego

Pierwsza część rozprawy poświęcona jest strukturodanowemu aspektowi ściągania (kontrakcji) krawędzi w grafach planarnych. Kontrakcja krawędzi jest jedną z fundamentalnych operacji na grafach. Załóżmy, że dany graf jest nieskierowany, a e jest jedną z jego krawędzi. Kontrakcja e polega na usunięciu tejże krawędzi z grafu, przy jednoczesnym scaleniu jej końców. Pojęcie ściągania krawędzi jest kluczowe w wielu spośród ważnych algorytmów grafowych, takich jak algorytm Edmondsa obliczający najliczniejsze skojarzenie w grafach ogólnych [19], czy algorytm Kargera obliczający minimalne cięcie [44].

Ściąganie krawędzi jest szczególnie ważne przy badaniu grafów planarnych, gdyż pewne ich

własności najłatwiej opisuje się przy użyciu tego pojęcia. Na przykład, wiadomo, że graf jest planarny dokładnie wtedy, gdy nie da się go przekształcić w K_5 lub $K_{3,3}$ za pomocą ciągu kontrakcji krawędzi, usunięć wierzchołków i usunięć krawędzi (zob. np. [17]). Dodatkowo, kontrakcja krawędzi zachowuje planarność grafu.

W związku z powyższym, chcielibyśmy mieć do dyspozycji strukturę danych, która wykonuje konktrakcje krawędzi na grafie wejściowym i w dalszym ciągu pozwala na szybki dostęp do podstawowych informacji o grafie, takich jak liczby sąsiadów poszczególnych wierzchołków czy relacja sąsiedztwa. Mimo że kontrakcja krawędzi jest pojęciem prostym koncepcyjnie, jej efektywna implementacja okazuje się nieoczywista. Dzieje się tak, ponieważ nie jest jasne, jak reprezentować listy sąsiedztwa poszczególnych wierzchołków, aby z jednej strony można było je szybko scalać, a z drugiej – równie efektywnie obsługiwać zapytania o stopnie i sąsiedztwo poszczególnych wierzchołków. Przy użyciu standardowych struktur danych (np. zrównoważonych drzew poszukiwań binarnych), możemy utrzymywać listy sąsiedztwa tak, aby zamortyzowany czas wykonania kontrakcji krawędzi był polilogarytmiczny. Jednakże, w niektórych algorytmach dla grafów planarnych właśnie ta operacja jest wąskim gardłem obliczeń.

Jako przykład rozważmy problem 5-kolorowania grafu planarnego. Istnieje bardzo prosty algorytm bazujący na kontrakcjach krawędzi [53], używający w zasadzie tylko znanego powszechnie faktu, że każdy graf planarny ma wierzchołek o stopniu co najwyżej 5. Mimo to, wszystkie znane liniowe algorytmy rozwiązujące ten problem używają znacznie bardziej zaawansowanych własności grafów planarnych [23, 53, 60]. Na przykład, w algorytmie Matuli i in. [53], kluczowy jest fakt, że każdy graf planarny ma wierzchołek o stopniu co najwyżej 4 lub wierzchołek o stopniu dokładnie 5, sąsiadujący z co najmniej czterema wierzchołkami, z których każdy ma stopnień nie większy niż 11. Podobnie, mimo że istnieje bardzo prosty algorytm obliczający minimalne drzewo rozpinające oparty na ściąganiu tanich krawędzi, algorytmy liniowe dla tego problemu na grafach planarnych wymagały do tej pory także innych metod [23, 51, 52].

Problem utrzymywania grafu planarnego podlegającego kontrakcjom krawędzi był już wcześniej podejmowany. W swojej książce, Klein i Mozes [46] pokazali (trochę ogólniejszą) strukturę danych utrzymującą podstawowe informacje o grafie, podczas gdy ten podlega kontrakcjom i usunięciom krawędzi. Zapytania o relację sąsiedztwa są w tej strukturze danych obsługiwane w pesymistycznym czasie stałym, a wykonanie kontrakcji wymaga zamortyzowanego czasu $O(\log n)$. Wynik ten opiera się na wcześniejszej pracy Brodala i Fagerberga [8], którzy pokazali, jak utrzymywać na dynamicznym grafie planarnym tzw. orientację o ograniczonym stopniu wyjściowym w zamortyzowanym czasie $O(\log n)$ na dodanie bądź usunięcie krawędzi grafu.

Gustedt [32] pokazał optymalne rozwiązanie dla problemu union-find w przypadku, gdy utrzymywane przez strukturę danych podzbiory w każdym momencie odpowiadają rozłącznym, spójnym podgrafom danego na początku grafu G. Innymi słowy, w tak zdefiniowanym problemie, dozwolone operacje łączenia zbiorów odpowiadają krawędziom danego grafu planarnego G, a na samą operację łączenia można patrzeć jak na kontrakcję odpowiedniej krawędzi.

Nasze wyniki

Pokazujemy strukturę danych umożliwiającą efektywne utrzymywanie podstawowych informacji o grafie planarnym podlegającym kontrakcjom. Dowolny ciąg kontrakcji jest obsługiwany w sumarycznym czasie liniowym, zakładając standardowy model word-RAM ze słowem maszynowym długości $\Omega(\log n)$. Po każdej kontrakcji, struktura dodatkowo wykrywa i zgłasza wszystkie nowo powstałe pętle i krawędzie wielokrotne. Zapytania o relację sąsiedztwa są obsługiwane w czasie stałym, a listy sąsiedztwa i stopnie poszczególnych wierzchołków są utrzymywanie jawnie. Nasza struktura danych może być użyta jako „czarna skrzynka" do implementacji algorytmów na grafach planarnych opisanych za pomocą kontrakcji krawędzi.

Jako przykład pokazujemy koncepcyjnie proste liniowe algorytmy obliczające 5-kolorowanie i minimalne drzewo rozpinające grafu planarnego.

Co ważniejsze, używając naszego wyniku, uzyskujemy szybsze, optymalne algorytmy dla kilku innych problemów na grafach planarnych. W szczególności, pokazujemy optymalny algorytm dla problemów dekrementalnej 2-spójności krawędziowej (zob. np. [30]), znajdowania unikalnych doskonałych skojarzeń [26] i obliczania maksymalnych 3-spójnych krawędziowo podgrafów [12].

Pierwszym krokiem naszej konstrukcji jest dekompozycja grafu na małe kawałki o (w przybliżeniu) logarytmicznym rozmiarze (za pomocą tzw. *r-podziałów* [24]). Nasz problem rozwiązujemy, mówiąc zgrubnie, budując strukturę danych rekurencyjnie dla każdego kawałka i osobno używając mniej wyrafinowanego podejścia dla podgrafu G indukowanego przez $o(n)$ wierzchołków występujących w co najmniej dwóch kawałkach jednocześnie (tzw. wierzchołków brzegowych). Takie podejście okazało się skuteczne przy uzyskiwaniu optymalnych rozwiązań dla problemów zbiorów rozłącznych [32] i dekrementalnej spójności [50] na grafach planarnych. W rzeczywistości, na problem strukturodanowy, który rozwiązujemy, można patrzeć jak na uogólnienie planarnego problemu union-find [32]. Jednakże, utrzymywanie dla każdej krawędzi e grafu jej obecnego stanu (tzn., informacji czy krawędź e stała się pętlą lub jedną z krawędzi wielokrotnych) podczas gdy kontrakcje postępują, a także obsługiwanie zapytań o relację sąsiedztwa bez użycia randomizacji, okazują się być poważnymi trudnościami technicznymi. Rozwikłanie tych trudności jest naszym głównym wkładem w tej części rozprawy.

3 Dekrementalna osiągalność

Druga część rozprawy poświęcona jest problemom dynamicznej osiągalności w grafach planarnych. W problemach *dynamicznej osiągalności* mamy dany (być może skierowany) graf G, podlegający zmianom zbioru krawędzi. Naszym celem jest zaprojektowanie struktury danych odpowiadającej na zapytania o istnienie ścieżki pomiędzy daną w zapytaniu parą wierzchołków $u, v \in V$.

Najczęściej rozważa się dwa warianty dynamicznej osiągalności. W wariancie *wszystkich par*, struktura danych ma obsługiwać zapytania pomiędzy dowolnymi parami wierzchołków. Ten wariant zwany jest także *dynamicznym domknięciem przechodnim*, gdyż ścieżka $u \to v$ istnieje w grafie G dokładnie wtedy, gdy uv jest krawędzią domknięcia przechodniego G.

Drugi wariant to problem dynamicznej osiągalności z pojedynczego źródła. W tym problemie wierzchołek źródłowy $s \in V$ jest ustalony od samego początku i dopuszcza się tylko zapytania o istnienie ścieżek z s do dowolnego wierzchołka $v \in V$.

Gdy ograniczymy się do grafów nieskierowanych, problem dynamicznej osiągalności jest nazywany problemem *dynamicznej spójności*. Zauważmy, że w przypadku nieskierowanym ścieżka z u do v istnieje w G wtedy i tylko wtedy, gdy istnieje też ścieżka z v do u.

Stan wiedzy

Problem dynamicznej osiągalności w ogólnych grafach skierowanych stanowi duże wyzwanie. Po pierwsze, wydaje się zdecydowanie trudniejszy obliczeniowo niż jego odpowiednik w grafach nieskierowanych. Dla grafów nieskierowanych znane są w pełni dynamiczne algorytmy dla wariantu wszystkich par, z polilogarytmicznymi czasami zmian i zapytań [36, 38, 71]. Dla grafów nieskierowanych, z drugiej strony, w większości scenariuszy (w wariantach jednego źródła bądź wszystkich par, dla modeli inkrementalnych, dekrementalnych i w pełni dynamicznych) najlepszy znany algorytm ma albo wielomianowy czas zmiany, albo wielomianowy czas zapytania. Jedynym znanym nam wyjątkiem jest problem inkrementalnej osiągalności z jednego źródła,

dla którego proste rozszerzenie algorytmu przeszukiwania w głąb [68] daje algorytm o stałym zamortyzowanym czasie zmiany.

Jedną z przyczyn tak wielkiej przepaści pomiędzy trudnością dynamicznej osiągalności w grafach nieskierowanych i skierowanych może być to, że spójne składowe (a więc i relację osiągalności) grafu nieskierowanego daje się łatwo obliczyć w czasie liniowym. Innymi słowy, w czasie liniowym można zbudować *statyczną* strukturę danych, która odpowiada na zapytania o osiągalność pomiędzy dowolnymi parami wierzchołków w czasie stałym. Z drugiej strony, najlepszy znany algorytm obliczający domknięcie przechodnie (a więc relację osiągalności) w grafie skierowanym działa w czasie $\widetilde{O}(\min(n^\omega, nm)) = \widetilde{O}(n^2)$[1] [11, 59].

Jak dotąd, najlepsze znane ograniczenia złożoności czasowej dla algorytmów w pełni dynamicznej osiągalności wyglądają następująco. Jeśli chodzi o dynamiczne domknięcie przechodnie, istnieje kilka algorytmów o czasie zmiany $O(n^2)$ i czasie zapytania $O(1)$ [16, 61, 64]. Algorytmy te tak naprawdę utrzymują domknięcie przechodnie w sposób jawny. Istnieje także kilka w pełni dynamicznych algorytmów o lepszym czasie zmiany (lecz wciąż rzędu $\Omega(n)$) dla grafów rzadkich i czasie zapytania $o(n)$ (ale w dalszym ciągu wielomianowym od n) [62, 63, 64]. Dla wariantu z jednym źródłem, jedyny znany nietrywialny algorytm (inny niż „przelicz wszystko od początku") ma czas zmiany $O(n^{1.53})$ i czas zapytania $O(1)$ [64].

Algorytmy o całkowitym czasie zmian $O(nm)$ są znane zarówno dla inkrementalnego [39], jak i dekrementalnego [48, 62] domknięcia przechodniego. Warto zauważyć, że dla grafów rzadkich, te ograniczenia czasowe są (z dokładnością do czynników polilogatymicznych) takie same jak najlepsze ograniczenia dla statycznego problemu domknięcia przechodniego [11].

Wszystkie znane częściowo dynamiczne algorytmy dla osiągalności z jednego źródła utrzymują zbiór wierzchołków osiągalnych w sposób jawny. Jak wspomniano wcześniej, dla inkrementalnej osiągalności z jednego źródła znany jest algorytm optymalny (w sensie zamortyzowanym). Co ciekawe, pierwsze algorytmy z całkowitym czasem zmian $O(mn^{1-\epsilon})$ (gdzie $\epsilon > 0$) uzyskano dopiero niedawno [33, 34]. Najlepszy znany w tym momencie algorytm dla tego problemu został podany przez Chechik i in. [13] i ma oczekiwany całkowity czas zmian $\widetilde{O}(m\sqrt{n})$.

Dynamiczna osiągalność była także dotychczas badana dla grafów planarnych. Diks i Sankowski [18] pokazali w pełni dynamiczny algorytm dla domknięcia przechodniego o czasie zmiany i zapytania $\widetilde{O}(\sqrt{n})$, działający przy założeniu, że rysunek grafu jest ustalony, a wstawiane krawędzie mogą łączyć tylko wierzchołki dzielące ze sobą pewną ścianę rysunku. Łącki [48] pokazał algorytm utrzymujący silnie spójne składowe grafu planarnego podlegającego usunięciom krawędzi o całkowitym czasie zmiany $O(n\sqrt{n})$. Korzystające ze znanych redukcji, można ten algorytm przekształcić w dekrementalny algorytm osiągalności z jednego źródła, także o całkowitym czasie zmian $O(n\sqrt{n})$. Zauważmy, że dla grafów planarnych to ograniczenie czasowe jest równe najlepszemu ograniczeniu dla grafów ogólnych [13], z dokładnością do czynników polilogarytmicznych.

Nasze wyniki

Pokazujemy nowe algorytmy dla problemu dekrementalnej osiągalności w grafach planarnych.

Dla dekrementalnej osiągalności z jednego źródła, uzyskujemy prawie optymalny (z dokładnością do czynników logarytmicznych) algorytm jawnie utrzymujący zbiór wierzchołków osiągalnych ze źródła. Nasz algorytm ma całkowity czas zmian $O(n \log^2 n \log\log n)$. Wydaje się, że to pierwszy algorytm dynamiczny z polilogarytmicznym zamortyzowanym czasem zmiany dla skierowanych grafów planarnych, rozwiązujący nietrywialny problem osiągalności, dla którego, w przypadku grafów ogólnych, wciąż nie są znane podwielomianowe ograniczenia czasowe.

[1] Przez $\widetilde{O}(f(n))$ oznaczamy rząd wielkości $O(f(n)\,\mathrm{polylog}\,n)$.

Dla problemu dekrementalnego domknięcia przechodniego, uzyskujemy algorytm randomizowany. Dla dowolnie wybranego $t \in [1, n]$, nasz algorytm ma całkowity czas zmian $\widetilde{O}(n^2/t)$ i czas zapytania $\widetilde{O}(\sqrt{t})$. W szczególności, dla $t = n$, uzyskujemy algorytm o polilogarytmicznym czasie zmiany i czasie $\widetilde{O}(\sqrt{n})$ na zapytanie. Dla $t = n^{2/3}$ z kolei, otrzymujemy algorytm o czasie zmiany/zapytania $\widetilde{O}(n^{1/3})$. O ile nam wiadomo, to pierwszy dynamiczny algorytm na skierowanych grafach planarnych odpowiadający na zapytania pomiędzy dowolnymi parami wierzchołków i jednocześnie mający ograniczenia czasowe rzędu $O(n^{1/2-\epsilon})$, dla pewnego $\epsilon > 0$.

Jako efekt uboczny, uzyskujemy także dla grafów planarnych prawie liniowe algorytmy dla kilku statycznych i dekrementalnych problemów związanych z osiągalnością, które wcześniej rozważane były tylko dla grafów ogólnych i dla których nie były znane algorytmy prawie liniowe. Są to, między innymi, obliczanie maksymalnych 2-krawędziowo-spójnych podgrafów grafu skierowanego [12] i dekrementalne utrzymywanie zbioru tzw. silnych mostów grafu [29].

Przegląd użytych technik

We wszystkich naszych algorytmach w tej części rozprawy zaczynamy od zbudowania *rekurencyjnej dekompozycji* wejściowego grafu G. Jest to drzewiasta hierarchia podgrafów G (kawałków) uzyskana poprzez rekurencyjny podział G przy użyciu cyklicznych separatorów rozmiaru $O(\sqrt{n})$ [54]. Dekompozycje tego typu okazały się kluczowe w uzyskaniu kilku spośród prawie liniowych statycznych algorytmów dla grafów planarnych (np. [3, 6, 49]), jak również algorytmów dynamicznych (np. [18]).

Kolejno, dla każdego kawałka H dekompozycji utrzymujemy tylko pewną część domknięcia przechodniego H, indukowaną przez *wierzchołki brzegowe* ∂H tego kawałka. Wierzchołki brzegowe H definiujemy jako te, które H dzieli z pozostałą częścią $G - H$ grafu G. Tym sposobem, sumaryczna ilość informacji, którą przechowujemy, jest prawie liniowa od n. Następnie dowodzimy, że wybrane części domknięcia przechodnich kawałków H mogą być obliczane indukcyjnie na podstawie samych domknięć przechodnich przechowywanych w kawałkach-dzieciach H. Podobnie, cała wymagana informacja o osiągalności w H może być *utrzymywana* indukcyjnie za pomocą dekrementalnego algorytmu domknięcia przechodniego, uruchomionego na domknięciach przechodnich dzieci H.

Aby otrzymać efektywny algorytm utrzymujący opisaną informację o osiągalności dla grafu podlegającego usunięciom krawędzi, analizujemy i wykorzystujemy strukturalne własności macierzy osiągalności pomiędzy wierzchołkami[2], które, mówiąc zgrubnie, leżą na małej liczbie ścian rysunku grafu (nieprzypadkowo jest to własność, którą posiadają zbiory wierzchołków brzegowych ∂H poszczególnych kawałków H dekompozycji). Spojrzenie macierzowe pozwala nam na uzyskanie użytecznych algorytmicznie własności, które niełatwo uchwycić, używając, poprzednio stosowanego do osiągalności w grafach planarnych, podejścia tzw. *ścieżki dzielącej* [18, 67, 69]. Kolejno pokazujemy, że uzyskane własności mogą być wykorzystane do bardzo efektywnego zasymulowania randomizowanego dekrementalnego algorytmu domknięcia przechodniego uzyskanego przez Bernsteina [2] tak, że całkowity czas zmian naszej rekurencyjnej struktury danych jest prawie liniowy od n.

Wreszcie, dowodzimy, że utrzymywana informacja o osiągalności wystarcza do obsługiwania zapytań o osiągalność pomiędzy dowolnymi parami wierzchołków w czasie $\widetilde{O}(\sqrt{n})$, a także do jawnego dekrementalnego utrzymywania silnie spójnych składowych G. Dzięki znanym redukcjom, to wystarcza do uzyskania problemu dekrementalnej osiągalności z jednego źródła z całkowitym czasem zmian bliskim liniowemu. Używające techniki Mozesa i Sommera [57], możemy zmniejszyć czas zapytania o istnienie ścieżki pomiędzy dowolnymi dwoma wierzchoł-

[2]Macierzą osiągalności zbioru wierzchołków S nazywamy podmacierz macierzy domknięcia przechodniego o wierszach i kolumnach odpowiadających wierzchołkom z S.

kami grafu do $\widetilde{O}(\sqrt{t})$, gdzie $t \in [1, n]$ jest dowolnie wybranym parametrem, kosztem zwiększenia całkowitego czasu zmian do $\widetilde{O}(n^2/t)$.

Zauważmy, że opisane powyżej metody prowadzą do algorytmów randomizowanych. W związku z tym, dowodzimy także, że używając dualności, możemy zredukować problem dekrementalnego utrzymywania silnie spójnych składowych do pewnego *inkrementalnego* problemu osiągalności, gdzie krawędzie grafu (ustalonego od początku) mogą być *włączane* w nieznanej przez nas kolejności. Taki problem można rozwiązać analogicznie, przy pomocy rekurencyjnej struktury danych, ale tym razem wymagana informacja o osiągalności w kawałkach musi być utrzymywana inkrementalnie. Stąd, możemy zastąpić dekrementalny algorytm Bernsteina koncepcyjnie prostszym algorytmem inkrementalnym, który na dodatek jest szybszy i deterministyczny. Dzięki temu otrzymujemy deterministyczny dekrementalny algorytm osiągalności z jednego źródła o całkowitym czasie zmian rzędu $O(n \log^2 n \log \log n)$.

4 Najkrótsze ścieżki w gęstych grafach odległości

Ostatni rozdział rozprawy poświęcony jest problemowi obliczania najkrótszych ścieżek w tzw. gęstych grafach odległości, podstawowemu narzędziu używanemu przy projektowaniu efektywnych algorytmów na grafach planarnych.

W przełomowej pracy [22], Fakcharoenphol i Rao wprowadzili ogólną ideę *gęstego grafu odległości* (ang. dense distance graph, w skrócie DDG). Załóżmy, że graf płaski G ma na krawędziach nieujemne wagi. Oznaczmy przez U pewien zbiór jego „brzegowych" wierzchołków rozlokowanych na pewnej, ograniczonej przez stałą, liczbie ścian G. Grafy z tego typu „ładnym" brzegiem najczęściej pojawiają się po podziale grafu płaskiego na dwie części przy użyciu separatora będącego cyklem. Na przykład, używając separatora Millera [54], można podzielić n-wierzchołkowy, striangulowany graf płaski H na dwa podgrafy $H_{\text{wewn.}}$ i $H_{\text{zewn.}}$ (wewnątrz i na zewnątrz cyklu) tak, aby (i) $H_{\text{wewn.}} \cup H_{\text{zewn.}} = H$, (ii) $H_{\text{wewn.}}$ i $H_{\text{zewn.}}$ miały po co najwyżej $\frac{2}{3}n + O(\sqrt{n})$ wierzchołków, (iii) zbiór $U = V(H_{\text{wewn.}}) \cap V(H_{\text{zewn.}})$ był wielkości $O(\sqrt{n})$ i leżał zarówno na jednej ścianie podgrafu $H_{\text{wewn.}}$, jak i na jednej ścianie podgrafu $H_{\text{zewn.}}$.

Dla takiego grafu z wyróżnionym zbiorem U, definiujemy *klikę odległości* $\mathrm{DC}(G)$ jako pełny ważony graf na wierzchołkach U taki, że waga krawędzi uv w $\mathrm{DC}(G)$ jest równa długości najkrótszej ścieżki z u do v w G. Wreszcie, gęstym grafem odległości nazywamy sumę pewnej liczby q, potencjalnie nie związanych ze sobą, klik odległości $\mathrm{DC}(G_1), \ldots, \mathrm{DC}(G_q)$.

Fakcharoenphol i Rao [22] zaproponowali efektywną implementację algorytmu Dijkstry (później nazwaną *FR-Dijkstrą*) do obliczania najkrótszych ścieżek w gęstych grafach odległości. Ich algorytm używa czasu $O(b \log^2 n)$ na każdą klikę odległości o b wierzchołkach, mimo iż taka klika ma b^2 krawędzi. Tutaj, n oznacza sumaryczną liczbę wierzchołków gęstego grafu odległości. Podczas gdy algorytm Dijkstry używa kolejki priorytetowej do utrzymywania oszacowań odległości i wyłuskiwania nieodwiedzonego wierzchołka o najmniejszym oszacowaniu, znacznie bardziej wyszukana struktura danych użyta jest w algorytmie FR-Dijkstra. Struktura ta jest w stanie zrelaksować wiele krawędzi w jednym kroku, korzystając z obserwacji, że pewne podmacierze macierzy wag kliki odległości stanowią tak zwane *macierze Monge'a* [70].

Fakcharoenphol i Rao pierwotnie użyli swojej implementacji do rekurencyjnej budowy gęstego grafu odległości, który potem użyli do rozwiązania problemu najkrótszych ścieżek z jednego źródła z ujemnymi wagami w grafach planarnych w czasie prawie liniowym. Jednakże, zastosowania algorytmu FR-Dijkstra okazały się później dużo szersze i algorytm ten stał się podstawowym narzędziem używanym do uzyskania wielu spośród przełomowych wyników dla grafów planarnych w ostatnich latach. Poniżej pokrótce opisujemy najważniejsze z tych odkryć.

Gęste grafy odległości i FR-Dijkstra były kluczowe do pokonania długo utrzymującej się bariery czasowej $O(n \log n)$ dla problemów obliczania minimalnych s, t-cięć [41] w nieskierowa-

nych grafach planarnych i globalnych minimalnych cięć, zarówno w nieskierowanych [47], jak i skierowanych [55] grafach planarnych. Borradaile i in. [6] podali strukturę danych o prawie liniowym czasie konstrukcji, odpowiadającą na zapytania o minimalne s,t-cięcie w ważonym nieskierowanym grafie planarnym (dla dowolnie wybranych wierzchołków s,t) w czasie polilogarytmicznym. Wynik ten został później uogólniony na grafy na powierzchniach o genusie ograniczonym przez stałą [3], co pokazuje, że FR-Dijkstra może się także przydać w algorytmach na grafach z bardziej ogólnych klas.

Najbardziej wyrafinowane, jak dotąd, zastosowania FR-Dijkstry to prawdopodobnie algorytmy związane z obliczaniem maksymalnych przepływów w skierowanych grafach planarnych. Borradaile i in. [5] pokazali prawie liniowy algorytm dla maksymalnego przepływu w przypadku wielu źródeł i ujść, a co za tym idzie, prawie liniowy algorytm dla maksymalnych skojarzeń w przypadku dwudzielnym. Później, Łącki i in. [49] podali prawie liniowy algorytm obliczania wartości maksymalnego przepływu z ustalonego źródła do wszystkich pozostałych wierzchołków.

Ostatnio, Asathulla i in. [1] użyli (między innymi) algorytmu FR-Dijkstra do przełamania bariery czasu $O(n^{3/2})$ dla planarnego problemu przydziału w przypadku całkowitoliczbowych kosztów. Cabello [9] pokazał pierwszy istotnie szybszy niż kwadratowy algorytm obliczający średnicę ważonego grafu planarnego. Mimo iż ten wynik bazuje głównie na nowym pomyśle tzw. addytywnie ważonych diagramów Voronoia dla grafów planarnych, gęste grafy odległości i FR-Dijkstra są w nim także istotnym elementem.

Nie mniej ważny jest fakt, że FR-Dijkstra była kluczem do uzyskania wszystkich istniejących *dokładnych* dynamicznych algorytmów o podliniowych czasach zmian i zapytań dla najkrótszych ścieżek, maksymalnych przepływów i minimalnych cięć w grafach planarnych [22, 41, 42, 45, 47].

Podsumowując, gęste grafy odległości mają ogromne znaczenie w algorytmice grafów planarnych i stąd uzyskanie możliwie najszybszych algorytmów obliczających je i manipulujących nimi jest ważnym kierunkiem badań. Chociaż szybszy algorytm (w stosunku do rekurencyjnej metody podanej przez [22]), działający w czasie $O((|V| + |U|^2) \log n)$, został zaproponowany dla problemu obliczania kliki odległości [45], poprawienie samego algorytmu FR-Dijkstra (obliczania najkrótszych ścieżek w gęstym grafie odległości) okazało się być dużym wyzwaniem i w ogólnym przypadku nie udało się znaleźć szybszego algorytmu [22].[3]

Nasze wyniki

Pokazujemy algorytm obliczający najkrótsze ścieżki z jednego źródła w gęstym grafie odległości o czasie działania $O\left(b\frac{\log^2 n}{\log^2 \log n}\right)$ na klikę odległości o b wierzchołkach.

Nasz wynik implikuje natychmiastowe poprawienie, o czynnik $O(\log^2 n \log n)$, najlepszych oszacowań złożoności czasowej dla wielu problemów na skierowanych grafach planarnych, takich jak maksymalny przepływ o wielu źródłach i ujściach, maksymalne skojarzenie dwudzielne [5] i maksymalny przepływ z jednego źródła do wielu ujść [49], dla których znane były algorytmy o złożoności $O(n \log^3 n)$, a więc już bliskie liniowym. Osiągamy także nieznaczne przyspieszenia dynamicznych algorytmów dla najkrótszych ścieżek i maksymalnych przepływów [41, 42, 45]. Mówiąc bardziej ogólnie, uzyskujemy lepsze oszacowania czasowe złożoności wszystkich tych problemów na grafach planarnych, dla których jedynym wąskim gardłem najlepszego znanego algorytmu jest obliczanie najkrótszych ścieżek w gęstym grafie odległości.

Przegląd użytych technik

Zajmujemy się problemem znajdowania najkrótszych ścieżek w gęstych grafach odległości z czysto stukturodanowego punktu widzenia. Ogólnie rzecz biorąc, zamiast zupełnie nowego

[3] Nieznacznie lepsze ograniczenia czasowe pokazano tylko w pewnym szczególnym przypadku [56].

algorytmu najkrótszych ścieżek, pokazujemy nową strukturę danych utrzymującą oszacowania odległości i wyłuskującą wierzchołki o najmniejszym oszacowaniu w zamortyzowanym czasie $O\left(\frac{\log^2 b}{\log^2 \log b}\right)$ (podczas gdy czas ten był rzędu $O(\log^2 b)$ w [22]).

W pracy [22], macierz wag kliki odległości jest najpierw dzielona na *kwadratowe* macierze Monge'a, z których każda odpowiada podzbiorowi krawędzi kliki. Dla każdej takiej macierzy używana jest osobna struktura danych do relaksacji odpowiednich krawędzi i wyłuskiwania oszacowań wyindukowanych przez te relaksacje. Przypomnijmy, że w przypadku algorytmy Dijkstry, przyspieszenie z czasu $O(m \log n)$ do czasu $O(m + n \log n)$ można uzyskać zauważając, że relaksacja pojedynczej krawędzi jest w pewnym sensie mniej kosztowna niż wyłuskanie wierzchołka o najmniejszym oszacowaniu. W związku z tym, można zastosować tzw. kopiec Fibonacciego [25] zamiast zwykłego kopca binarnego. Pokazujemy, że w przypadku struktury danych podanej w [22] do obsługi macierzy Monge'a, sytuacja jest w pewnym sensie odwrotna: wyłuskiwanie najmniejszych ograniczeń można zaimplementować szybciej niż relaksacje wielu krawędzi naraz. Korzystamy z tego faktu proponujemy inny niż w [22], ukierunkowany podział macierzy wag kliki odległości na *prostokątne* (a więc niekoniecznie kwadratowe) macierze Monge'a. Podczas gdy w pracy [22] podział kliki odległości odbywa się zgodnie z naturalnym pomysłem dzielenia ściany zawierającej wierzchołki kliki na połowy, nasz schemat podziału jest podyktowany możliwością wykorzystania różnicy w kosztach przetwarzania pojedynczego wiersza i pojedynczej kolumny macierzy Monge'a.

5 Artykuły składające się na tę rozprawę

Wstępne wersje treści tej rozprawy zawarte były w następujących publikacjach konferencyjnych.

- *Contracting a Planar Graph Efficiently*, razem z Jacobem Holmem, Giuseppe F. Italiano, Jakubem Łąckim, Evą Rotenberg i Piotrem Sankowskim, opublikowana na konferencji ESA 2017 [37].
- *Decremental Single-Source Reachability in Planar Digraphs*, razem z Giuseppe F. Italiano, Jakubem Łąckim i Piotrem Sankowskim, opublikowana na konferencji STOC 2017 [40].
- *Decremental Transitive Closure and Shortest Paths for Planar Digraphs and Beyond*, opublikowana na konferencji SODA 2018 [43].
- *Improved Bounds for Shortest Paths in Dense Distance Graphs*, razem z Pawłem Gawrychowskim, opublikowana na konferencji ICALP 2018 [28].

Literatura

[1] Mudabir Kabir Asathulla, Sanjeev Khanna, Nathaniel Lahn, and Sharath Raghvendra. A faster algorithm for minimum-cost bipartite perfect matching in planar graphs. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 457–476, 2018.

[2] Aaron Bernstein. Maintaining shortest paths under deletions in weighted directed graphs. *SIAM J. Comput.*, 45(2):548–574, 2016.

[3] Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA*, pages 22:1–22:16, 2016.

[4] Glencora Borradaile and Philip N. Klein. An $O(n \log n)$ algorithm for maximum st-flow in a directed planar graph. *J. ACM*, 56(2):9:1–9:30, 2009.

[5] Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.*, 46(4):1280–1303, 2017.

[6] Glencora Borradaile, Piotr Sankowski, and Christian Wulff-Nilsen. Min st-cut oracle for planar graphs with near-linear preprocessing time. *ACM Trans. Algorithms*, 11(3):16:1–16:29, 2015.

[7] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, 2004.

[8] Gerth Stølting Brodal and Rolf Fagerberg. Dynamic representation of sparse graphs. In *Algorithms and Data Structures, 6th International Workshop, WADS '99, Vancouver, British Columbia, Canada, August 11-14, 1999, Proceedings*, pages 342–351, 1999.

[9] Sergio Cabello. Subquadratic algorithms for the diameter and the sum of pairwise distances in planar graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2143–2152, 2017.

[10] Sergio Cabello, Erin W. Chambers, and Jeff Erickson. Multiple-source shortest paths in embedded graphs. *SIAM J. Comput.*, 42(4):1542–1571, 2013.

[11] Timothy M. Chan. All-pairs shortest paths with real weights in $O\ (n\ ^3/\log\ n)$ time. *Algorithmica*, 50(2):236–243, 2008.

[12] Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F. Italiano, Veronika Loitzenbauer, and Nikos Parotsidis. Faster algorithms for computing maximal 2-connected subgraphs in sparse directed graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1900–1918, 2017.

[13] Shiri Chechik, Thomas Dueholm Hansen, Giuseppe F. Italiano, Jakub Łącki, and Nikos Parotsidis. Decremental single-source reachability and strongly connected components in $\widetilde{O}(m\sqrt{n}\,)$ total update time. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 315–324, 2016.

[14] Michael B. Cohen, Aleksander Madry, Piotr Sankowski, and Adrian Vladu. Negative-weight shortest paths and unit capacity minimum cost flow in õ $(m^{10/7}\ \log\ W)$ time (extended abstract). In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 752–771, 2017.

[15] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009.

[16] Camil Demetrescu and Giuseppe F. Italiano. Mantaining dynamic matrices for fully dynamic transitive closure. *Algorithmica*, 51(4):387–427, 2008.

[17] Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

[18] Krzysztof Diks and Piotr Sankowski. Dynamic plane transitive closure. In *Algorithms - ESA 2007, 15th Annual European Symposium, Eilat, Israel, October 8-10, 2007, Proceedings*, pages 594–604, 2007.

[19] Jack Edmonds. Paths, trees and flowers. *Canadian Journal of Mathematics*, pages 449–467, 1965.

[20] David Eppstein and Michael T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *16th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2008, November 5-7, 2008, Irvine, California, USA, Proceedings*, page 16, 2008.

[21] Jeff Erickson. Maximum flows and parametric shortest paths in planar graphs. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 794–804, 2010.

[22] Jittat Fakcharoenphol and Satish Rao. Planar graphs, negative weight edges, shortest paths, and near linear time. *J. Comput. Syst. Sci.*, 72(5):868–889, 2006.

[23] Greg N. Frederickson. On linear-time algorithms for five-coloring planar graphs. *Inf. Process. Lett.*, 19(5):219–224, 1984.

[24] Greg N. Frederickson. Fast algorithms for shortest paths in planar graphs, with applications. *SIAM J. Comput.*, 16(6):1004–1022, 1987.

[25] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987.

[26] Harold N. Gabow, Haim Kaplan, and Robert Endre Tarjan. Unique maximum matching algorithms. *J. Algorithms*, 40(2):159–183, 2001.

[27] Pawel Gawrychowski, Haim Kaplan, Shay Mozes, Micha Sharir, and Oren Weimann. Voronoi diagrams on planar graphs, and computing the diameter in deterministic $\tilde{O}(n^{5/3})$ time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 495–514, 2018.

[28] Pawel Gawrychowski and Adam Karczmarz. Improved bounds for shortest paths in dense distance graphs. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 61:1–61:15, 2018.

[29] Loukas Georgiadis, Thomas Dueholm Hansen, Giuseppe F. Italiano, Sebastian Krinninger, and Nikos Parotsidis. Decremental data structures for connectivity and dominators in directed graphs. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 42:1–42:15, 2017.

[30] Dora Giammarresi and Giuseppe F. Italiano. Decremental 2- and 3-connectivity on planar graphs. *Algorithmica*, 16(3):263–287, 1996.

[31] Andrew V. Goldberg. Scaling algorithms for the shortest paths problem. *SIAM J. Comput.*, 24(3):494–504, 1995.

[32] Jens Gustedt. Efficient union-find for planar graphs and other sparse graph classes. *Theor. Comput. Sci.*, 203(1):123–141, 1998.

[33] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Sublinear-time decremental algorithms for single-source reachability and shortest paths on directed graphs. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 674–683, 2014.

[34] Monika Henzinger, Sebastian Krinninger, and Danupon Nanongkai. Improved algorithms for decremental single-source reachability on directed graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, Proceedings, Part I*, pages 725–736, 2015.

[35] Monika Rauch Henzinger, Philip N. Klein, Satish Rao, and Sairam Subramanian. Faster shortest-path algorithms for planar graphs. *J. Comput. Syst. Sci.*, 55(1):3–23, 1997.

[36] Jacob Holm, Kristian de Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *J. ACM*, 48(4):723–760, 2001.

[37] Jacob Holm, Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, Eva Rotenberg, and Piotr Sankowski. Contracting a planar graph efficiently. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, pages 50:1–50:15, 2017.

[38] Shang-En Huang, Dawei Huang, Tsvi Kopelowitz, and Seth Pettie. Fully dynamic connectivity in $O(\log n(\log \log n)^2)$ amortized expected time. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete 16-19*, pages 510–520, 2017.

[39] Giuseppe F. Italiano. Amortized efficiency of a path retrieval data structure. *Theor. Comput. Sci.*, 48(3):273–281, 1986.

[40] Giuseppe F. Italiano, Adam Karczmarz, Jakub Łącki, and Piotr Sankowski. Decremental single-source reachability in planar digraphs. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1108–1121, 2017.

[41] Giuseppe F. Italiano, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Improved algorithms for min cut and max flow in undirected planar graphs. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 313–322, 2011.

[42] Haim Kaplan, Shay Mozes, Yahav Nussbaum, and Micha Sharir. Submatrix maximum queries in monge matrices and partial monge matrices, and their applications. *ACM Trans. Algorithms*, 13(2):26:1–26:42, 2017.

[43] Adam Karczmarz. Decremental transitive closure and shortest paths for planar digraphs and beyond. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 73–92, 2018.

[44] David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA.*, pages 21–30, 1993.

[45] Philip N. Klein. Multiple-source shortest paths in planar graphs. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2005, Vancouver, British Columbia, Canada, January 23-25, 2005*, pages 146–155, 2005.

[46] Philip N. Klein and Shay Mozes. Optimization algorithms for planar graphs, 2017.

[47] Jakub Łącki, and Piotr Sankowski and. Min-cuts and shortest cycles in planar graphs in $O(n \log \log n)$ time. In *Algorithms - ESA 2011 - 19th Annual European Symposium, Saarbrücken, Germany, September 5-9, 2011. Proceedings*, pages 155–166, 2011.

[48] Jakub Łącki. Improved deterministic algorithms for decremental reachability and strongly connected components. *ACM Trans. Alg.*, 9(3):27:1–27:15, 2013.

[49] Jakub Łącki and Yahav Nussbaum and Piotr Sankowski and Christian Wulff-Nilsen. Single source - all sinks max flows in planar digraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 599–608, 2012.

[50] Jakub Łącki and Piotr Sankowski. Optimal decremental connectivity in planar graphs. *Theory Comput. Syst.*, 61(4):1037–1053, 2017.

[51] Martin Mareš. Two linear time algorithms for mst on minor closed graph classes. *Archivum mathematicum*, 40(3):315–320, 2002.

[52] Tomomi Matsui. The minimum spanning tree problem on a planar graph. *Discrete Applied Mathematics*, 58(1):91–94, 1995.

[53] David W. Matula, Yossi Shiloach, and Robert E. Tarjan. Two linear-time algorithms for five-coloring a planar graph. Technical report, Stanford University, Stanford, CA, USA, 1980.

[54] Gary L. Miller. Finding small simple cycle separators for 2-connected planar graphs. *J. Comput. Syst. Sci.*, 32(3):265–279, 1986.

[55] Shay Mozes, Kirill Nikolaev, Yahav Nussbaum, and Oren Weimann. Minimum cut of directed planar graphs in $O(n \log \log n)$ time. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 477–494, 2018.

[56] Shay Mozes, Yahav Nussbaum, and Oren Weimann. Faster shortest paths in dense distance graphs, with applications. *Theor. Comput. Sci.*, 711:11–35, 2018.

[57] Shay Mozes and Christian Sommer. Exact distance oracles for planar graphs. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 209–222, 2012.

[58] Shay Mozes and Christian Wulff-Nilsen. Shortest paths in planar graphs with real lengths in $O(n\log^2 n/\log\log n)$ time. In *Algorithms - ESA 2010, 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part II*, pages 206–217, 2010.

[59] J. Ian Munro. Efficient determination of the transitive closure of a directed graph. *Inf. Process. Lett.*, 1(2):56–58, 1971.

[60] Neil Robertson, Daniel P. Sanders, Paul D. Seymour, and Robin Thomas. Efficiently four-coloring planar graphs. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 571–575, 1996.

[61] Liam Roditty. A faster and simpler fully dynamic transitive closure. *ACM Trans. Algorithms*, 4(1):6:1–6:16, 2008.

[62] Liam Roditty and Uri Zwick. Improved dynamic reachability algorithms for directed graphs. *SIAM J. Comput.*, 37(5):1455–1471, 2008.

[63] Liam Roditty and Uri Zwick. A fully dynamic reachability algorithm for directed graphs with an almost linear update time. *SIAM J. Comput.*, 45(3):712–733, 2016.

[64] Piotr Sankowski. Dynamic transitive closure via dynamic matrix inverse (extended abstract). In *45th Symposium on Foundations of Computer Science (FOCS 2004), 17-19 October 2004, Rome, Italy, Proceedings*, pages 509–517, 2004.

[65] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Number t. 1 in Algorithms and Combinatorics. Springer, 2003.

[66] Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. *J. Comput. Syst. Sci.*, 26(3):362–391, 1983.

[67] Sairam Subramanian. A fully dynamic data structure for reachability in planar digraphs. In *Algorithms - ESA '93, First Annual European Symposium, Bad Honnef, Germany, September 30 - October 2, 1993, Proceedings*, pages 372–383, 1993.

[68] Robert Endre Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[69] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *J. ACM*, 51(6):993–1024, 2004.

[70] Wikipedia. Monge array — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Monge_array`, 2018. [online; accessed 02-11-2018].

[71] Christian Wulff-Nilsen. Faster deterministic fully-dynamic graph connectivity. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1757–1769, 2013.