Nowe techniki stosowane przy rozwiązywaniu wybranych problemów NP-trudnych

autoreferat rozprawy doktorskiej

Marcin Pilipczuk

1 Wstęp

Problemy NP-trudne pojawiają się w praktycznych zastosowaniach często i, mimo że teoretycznie najprawdopodobniej nie da się ich rozwiązać efektywnie, naukowcy próbują sobie z nimi radzić. Najbardziej znaną metodą, i jedną z najstarszych, metod rozwiązywania problemów NP-trudnych jest aproksymacja: poszukujemy rozwiązania nie optymalnego, ale bliskiego optymalnemu. Niestety, badania związane z twierdzeniem PCP w latach 90. ubiegłego wieku, i hipotezą Unique Games w ostatnim dziesięcioleciu pokazały, że dla wielu problemów prawdopodobnie nie istnieją szybkie algorytmy o satysfakcjonującym współczynniku aproksymacji. W związku z tym należy poszukiwać innych metod radzenia sobie z problemami NP-trudnymi, w tym metod bardziej efektywnego rozwiązywania ich *optymalnie*. W mojej pracy doktorskiej zajmuję się dwoma takimi podejściami: algorytmami umiarkowanie wykładniczymi (ang. *moderately-exponential algorithms*) oraz złożonością parametryzowaną (ang. *parameterized complexity*).

2 Algorytmy umiarkowanie wykładnicze

Rozważmy problem komiwojażera: mając daną metrykę na n wierzchołkach, znaleźć najkrótszy cykl odwiedzający każdy wierzchołek dokładnie raz. W wersji decyzyjnej, w której pytamy o cykl krótszy niż zadana wartość, jest to problem NP-zupełny. Jak każdy problem należący do klasy NP, można go rozwiązać sprawdzając wszystkie możliwe rozwiązania (tzw. świadków, lub, inaczej ujmując, wszystkie ścieżki obliczeniowe niedeterministycznej maszyny Turinga rozwiązującej ten problem). W przypadku problemu komiwojażera, naturalnym świadkiem jest permutacja wierzchołków, w jakiej odwiedzamy je w poszukiwanym cyklu; rozwiązanie brutalne ma więc złożoność

$O^\star(n!)$ (notacja $O^\star$ pomija czynniki zależne wielomianowo od rozmiaru wejścia, gdyż są one zaniedbywalne przy wykładniczych czynnikach). Jak jednak pokazali już w 1962 roku Held i Karp, prosty algorytm programowania dynamicznego osiąga istotnie lepszą złożoność czasową $O^\star(2^n)$ [20].

Problem komiwojażera jest jednym z wielu przykładów problemów NP-zupełnych, które można rozwiązać dużo szybciej niż przez naiwną enumerację wszystkich możliwych rozwiązań. Okazuje się więc, że wyrok NP-zupełności, który w klasycznym ujęciu odkłada problem na półkę „nierozwiązywalne", nie przekreśla możliwości zastosowania różnych technik algorytmicznych w celu szybszego otrzymania optymalnego rozwiązania. Dziedzina algorytmów umiarkowanie wykładniczych, burzliwie rozwijająca się w ostatnim dziesięcioleciu, zajmuje się poszukiwaniem algorytmów rozwiązujących problemy NP-trudne szybciej niż przez naiwną enumerację możliwych rozwiązań.

Korzyści z tych badań są dwojakie. Po pierwsze, zauważmy, iż jeśli dla pewnego problemu poprawimy złożoność działania algorytmu z $O^\star(c_1^n)$ do $O^\star(c_2^n)$, możliwy zakres wielkości danych wejściowych (wartości n), dla których jesteśmy w stanie rozwiązać nasz problem w akceptowalnym czasie, powiększa się o stałą *multiplikatywną* $\log c_1 / \log c_2$, w przeciwieństwie do stałej *addytywnej* uzyskanej przez wykorzystanie kilkukrotnie szybszego komputera. Rozwój algorytmów umiarkowanie wykładniczych spowodował, że wiele problemów potrafimy rozwiązać już dla instancji co prawda małych, ale nie tylko trywialnych.

Po drugie, badania algorytmów umiarkowanie wykładniczych przyczyniły się do rozwoju w ostatnim dziesięcioleciu kilku ważnych technik algorytmicznych, takich jak „mierz i zwyciężaj" [15], algorytm szybkiego splotu podzbiorów [4] czy zastosowania probabilistycznego sprawdzania równości wielomianów [3]. Ponadto, opracowanie nowego algorytmu umiarkowanie wykładniczego dla danego problemu często wiąże się z nowymi, ciekawymi obserwacjami dotyczącymi tego problemu, które mogą mieć zastosowanie w innych obszarach algorytmiki.

Warto nadmienić, że gwałtowny rozwój dziedziny algorytmów umiarkowanie wykładniczych w ostatniej dekadzie został ukoronowany ukazaniem się książki autorstwa Fomina i Kratscha [16].

Jednym z ciekawych zjawisk obserwowanych w rozważanej dziedzinie jest występowanie kilku barier złożoności czasowej lub pamięciowej, których pokonanie w przypadku jakiegoś problemu uznaje się za bardzo interesujący wynik. Tak jest w przypadku otrzymania pierwszych algorytmów pojedynczo wykładniczych (o złożoności $O(c^n)$ dla stałej c; liczba n oznacza jakąś miarę wielkości danych wejściowych, np. liczbę wierzchołków grafu), podwykładniczych (o złożoności $2^{o(n)}$), czy też o złożoności $O(c^n)$ dla pewnej

stałej $c < 2$. Ta ostatnia bariera, zwana *barierą* 2^n, choć może wydawać się na pierwszy rzut oka zaskakująca, po głębszej analizie okazuje się być bardzo naturalna: często naiwna enumeracja wszystkich możliwych rozwiązań sprowadza się do przejrzenia wszystkich podzbiórów zbioru wierzchołków grafu (lub innego zbioru obiektów w danych wejściowych), lub też prosty algorytm programowania dynamicznego oblicza wartości dla każdego takiego podzbioru.

W mojej rozprawie doktorskiej pokazuję trzy nowe algorytmy umiarkowanie wykładnicze, dla problemów Capacitated Dominating Set, Minimum Maximal Irredundant Set i pewnego problemu szeregowania zadań (oznaczanego jako $1|\text{prec}|\sum C_i$ w notacji Grahama). W każdym przypadku prezentowany algorytm jest pierwszym znanym algorytmem działającym w czasie szybszym niż $O^\star(2^n)$, przełamując *barierę* 2^n. Wszystkie te wyniki oparte są na nowych i ciekawych obserwacjach dotyczących rozważanego problemu.

2.1 Capacitated Dominating Set

W danym grafie nieskierowanym $G = (V, E)$, zbiór $X \subseteq V$ nazwiemy *dominującym*, jeśli każdy wierzchołek grafu jest w X lub ma sąsiada w X. Problem znalezienia najmniejszego zbioru dominującego w grafie jest jednym z najbardziej znanych problemów NP-zupełnych. Zauważmy, że naiwny algorytm dla tego problemu przegląda wszystkie możliwe podzbiory zbioru wierzchołków i działa w czasie $O^\star(2^n)$.

Pytanie, czy barierę 2^n można przełamać dla problemu znajdowania minimalnego zbioru dominującego był jednym z podstawowych problemów otwartych w dziedzinie algorytmów umiarkowanie wykładniczych na początku obecnego stulecia. Skutkiem ubocznym przełamania tej bariery było opracowanie techniki „mierz i zwyciężaj" [15], będącej zaawansowaną metodą mierzenia złożoności algorytmów rozgałęziających się (ang. *branching*). Technika ta jest obecnie powszechnie używanym narzędziem w dziedzinie algorytmów umiarkowanie wykładniczych.

Jednym z wariantów problemu znajdowania minimalnego zbioru dominującego jest tzw. *dominacja z ograniczeniami pojemności* (Capacitated Dominating Set). W danym grafie $G = (V, E)$ każdy wierzchołek $v \in V$ ma swoją pojemność $c(v) \in \{0, 1, 2, \ldots\}$, która oznacza, ile sąsiadów wierzchołek v może dominować. Formalnie, mówimy, że zbiór $X \subseteq V$ jest zbiorem dominującym z ograniczeniami pojemności, jeśli istnieje funkcja $f : V \setminus X \to X$ taka, że:

- dla każdego wierzchołka $u \in V \setminus X$, wierzchołek $f(u)$ jest sąsiadem u;

- dla każdego wierzchołka $v \in X$, zbiór $f^{-1}(v)$ ma co najwyżej $c(v)$ elementów.

Zauważmy, iż weryfikacja, czy zbiór $X \subseteq V$ jest zbiorem dominującym z ograniczeniami pojemności, wymaga znalezienia funkcji f, co jest wykonalne w czasie wielomianowym, choć wymaga zaawansowanych algorytmów, np. znajdowania maksymalnego skojarzenia w dowolnych grafach. Ponadto, nawet znając zbiór X, a patrząc na mały wycinek grafu, trudno jest przewidzieć jak będzie zachowywać się funkcja f na tym kawałku. Powoduje to, że algorytmy rozgałęziające się — bardzo skuteczne w przypadku problemu klasycznego zbioru dominującego — przestają działać w wariancie z pojemnościami, gdyż podejmowanie lokalnych, zachłannych decyzji jest bardzo utrudnione.

Motywowany tą trudnością, Johan van Rooij w 2008 roku postawił pytanie [14], czy można przełamać barierę 2^n dla problemu minimalnego zbioru dominującego w wariancie z pojemnościami (zauważmy, że naiwne rozwiązanie działa w czasie $O^\star(2^n)$, gdyż można w czasie wielomianowym zweryfikować, czy zbiór X jest zbiorem dominującym w wariancie z pojemnościami). W rozprawie doktorskiej odpowiadam na to pytanie twierdząco:

Twierdzenie 1. *Problem* Capacitated Dominating Set *można rozwiązać w czasie* $O^\star(\binom{n}{n/3}) = O(1.89^n)$ *i pamięci wielomianowej, gdzie* n *oznacza liczbę wierzchołków wejściowego grafu.*

Opracowany algorytm jest zaskakująco prosty i opiera się na obserwacji, że problem zbioru dominującego w wariancie z pojemnościami można rozwiązać w czasie wielomianowym w przypadku, gdy wszystkie pojemności wynoszą 0 lub 1, poprzez redukcję do problemu maksymalnego skojarzenia w grafie dowolnym. By otrzymać nasz algorytm, wystarczy zauważyć jeszcze, że w dowolnym rozwiązaniu $X \subseteq V$ rozważanego problemu, przy ustalonej funkcji f, zbiór $X_{f\geq 2} = \{v \in X : |f^{-1}(v)| \geq 2\}$ jest wielkości co najwyżej $n/3$. Iterując po wszystkich możliwych zbiorach $X_{f\geq 2}$, których jest co najwyżej $n\binom{n}{n/3}$, otrzymujemy żądaną złożoność czasową.

2.2 Minimum Maximal Irredundant Set

Innym wariantem problemu znajdowania minimalnego zbioru dominującego jest problem znajdowania minimalnego, maksymalnego w sensie zawierania, zbioru bez redundancji (Minimum Maximal Irredundant Set). Mówimy, że wierzchołek v *dominuje* wierzchołek u, jeśli $v = u$ lub u jest sąsiadem v. Zbiór $X \subseteq V$ nazwiemy *bez redundancji*, jeśli dla każdego wierzchołka $v \in X$ istnieje wierzchołek $u_v \in V$ będący jego *prywatnym ogrodem*,

tj. takim wierzchołkiem, że v dominuje u_v, ale żaden wierzchołek $X \setminus \{v\}$ nie dominuje u_v (być może $u_v = v$). Łatwo zauważyć, że każdy minimalny w sensie zawierania zbiór dominujący jest zbiorem bez redundancji: w przeciwnym razie, możemy usunąć ze zbioru dominującego wierzchołek bez prywatnego ogrodu, zmniejszając zbiór dominujący.

Problem znajdowania minimalnego, maksymalnego w sensie zawierania, zbioru bez redundancji, oraz pokrewny problem znajdowania maksymalnego zbioru bez redundancji, są NP-zupełne, i naiwne rozwiązanie przegląda wszystkie podzbiory zbioru wierzchołków, uzyskując złożoność czasową $O^\star(2^n)$ (zwróćmy uwagę, że sprawdzenie, czy zbiór $X \subseteq V$ jest bez redundancji można wykonać w czasie liniowym wprost z definicji). Pytanie, czy te dwa problemy można rozwiązać szybciej, przełamując barierę 2^n, zostało postawione przez Johana van Rooija w 2008 roku [14]. W naszej pracy prezentowanej na konferencji CIAC 2010 [11] odpowiadamy na oba te pytania twierdząco; w rozprawie doktorskiej opisuje dużo ciekawszy, pierwszy z tych algorytmów.

Twierdzenie 2. *Problem* Minimum Maximal Irredundant Set *można rozwiązać w czasie* $O(1.999965^n)$*, gdzie* n *oznacza liczbę wierzchołków wejściowego grafu.*

Choć stała w złożoności powyższego algorytmu nie jest daleka od 2, wartością tego wyniku jest samo przełamanie bariery 2^n oraz pokazanie strukturalnych własności rozważanego problemu, prowadzących do ograniczenia przestrzeni możliwych rozwiązań.

W kilku zdaniach postaram się zaszkicować ideę prezentowanego algorytmu. Ustalmy stałą $\alpha < 1/2$. Jeśli przeglądając podzbiory zbioru wierzchołków o rozmiarze nie większym niż αn natrafimy na zbiór dominujący, natrafimy też na zbiór dominujący minimalny w sensie zawierania, a więc i na maksymalny w sensie zawierania zbiór bez redundancji. Takie poszukiwania mają złożoność czasową $O^\star(\binom{n}{\alpha n}) = O^\star(c_\alpha^n)$ dla pewnej stałej $c_\alpha < 2$ zależnej tylko od α. Jeśli zaś graf nie zawiera zbioru dominującego o rozmiarze αn, ma on dość specyficzną strukturę: pokazujemy, że musi on zawierać zbiór co najmniej $\varepsilon_\alpha n$ wierzchołków, o stopniu co najwyżej d_α i rozłącznych sąsiedztwach (stałe $\varepsilon_\alpha > 0$ i d_α zależą tylko od α). Zauważmy, że zbiór bez redundancji nie może zawierać wierzchołka wraz z całym jego sąsiedztwem (o ile nie jest to szczególny przypadek wierzchołka izolowanego). Gdy mamy $\varepsilon_\alpha n$ rozłącznych sąsiedztw o co najwyżej d_α wierzchołach, liczba zbiorów, które algorytm naiwny musi obejrzeć jest ograniczona przez

$$2^n \cdot \left(\frac{2^{d_\alpha+1} - 1}{2^{d_\alpha}} \right)^{\varepsilon_\alpha n} = b_\alpha^n$$

dla pewnego $b_\alpha < 2$ zależnego tylko do α.

2.3 Problem $1|\mathbf{prec}|\sum C_i$

Problematyka szeregowania zadań na jednej lub wielu maszynach jest jednym ze starszych, i głęboko badanych działów informatyki. Wiele problemów napotykanych w tej dziedzinie okazuje się NP-zupełnych. W mojej rozprawie doktorskiej zajmuję się jednym szczególnym zagadnieniem: szeregowaniem zadań z zależnościami na jednej maszynie ($1|\text{prec}|\sum C_i$ w notacji Grahama). W tym problemie mamy dany zbiór V zawierający n zadań, częściowy porządek $\leq$ na V, oraz, dla każdego zadania $v \in V$, jego czas wykonania $t(v) \geq 0$. Naszym celem jest takie uszeregowanie zadań (podanie bijekcji $\sigma : V \to \{1, 2, \ldots, n\}$), które by spełniało zależności między zadaniami (jeśli $u < v$, to u należy wykonać przed v, czyli $\sigma(u) < \sigma(v)$) oraz minimalizowało sumaryczny oczekiwania na wykonanie zadań:

$$T(\sigma) = \sum_{v \in V} \sum_{u:\sigma(u) \leq \sigma(v)} t(u).$$

Innymi słowy, dla każdego zadania v koszt wykonania tego zadania jest zdefiniowany jako czas, jaki minął od początku wykonywania wszystkich zadań, do końca wykonywania zadania v; całkowity koszt uszeregowania zadań to suma kosztów poszczególnych zadań.

Przekształćmy nieznacznie formułę na $T(\sigma)$:

$$T(\sigma) = \sum_{v \in V} \sum_{u:\sigma(u) \leq \sigma(v)} t(u) = \sum_{u \in V} (n - \sigma(u) + 1) t(u).$$

Tak przekształconą formułę można wprost zastosować w algorytmie programowania dynamicznego. Zauważmy, że jeśli wykonaliśmy już zadania $X \subseteq V$, to ważne jest dla nas tylko *które* zadania zostały wykonane, a nie *w jakiej kolejności*. Zdefiniujmy więc koszt częściowego uporządkowania (bijekcji) $\sigma_X : X \to \{1, 2, \ldots, |X|\}$ jako:

$$T(\sigma_X) = \sum_{u \in X} (n - \sigma(u) + 1) t(u).$$

Przez $M[X]$ oznaczmy (jedno z być może wielu) częściowe uporządkowanie σ_X zbioru $X \subseteq V$ takie, które minimalizuje $T(\sigma_X)$. Łatwo widać, że $M[X]$ można wyliczyć w czasie wielomianowym korzystając z wartości $M[X \setminus \{v\}]$ dla $v \in X$, zaś wartość $M[V]$ jest poszukiwanym przez nas uszeregowaniem.

Powyżej opisany algorytm programowania dynamicznego działa w czasie i pamięci $O^\star(2^n)$. Jedną z trudności w przyspieszeniu powyższego algorytmu jest obecność dowolnych, rzeczywistych wag (czasów działania zadań). Podobną sytuację napotykamy w przypadku problemu komiwojażera: w 2010 roku Björklund pokazał algorytm stwierdzający, czy dany nieskierowany graf ma cykl Hamiltona, działający w czasie $O^\star(1.66^n)$ [3], lecz jego podejście nie uogólnia się ani do problemu komiwojażera, ani nawet do przypadku grafów skierowanych. Problem komiwojażera (i wiele prokrewnych problemów z wagami) można rozwiązać przez proste programowanie dynamiczne w czasie $O^\star(2^n)$. Prawdopodobnie to zjawisko motywowało Woegingera, który w 2004 roku postawił pytanie [27], czy problem $1|\text{prec}|\sum C_i$ jest równie trudny jak problem komiwojażera, czy też można łatwiej przełamać barierę 2^n.

W mojej rozprawie doktorskiej odpowiadam na to pytanie, dowodząc następującego twierdzenia.

Twierdzenie 3. *Problem $1|prec|\sum C_i$ można rozwiązać w czasie $O((2-5\cdot 10^{-16})^n)$, gdzie n jest liczbą zadań w danych wejściowych.*

Wartością tego wyniku jest więc sam fakt przełamania bariery 2^n, jak i pokazania kilku ciekawych obserwacji dotyczących problemu $1|\text{prec}|\sum C_i$; samo przyspieszenie działania algorytmu jest pomijalne.

Opracowany algorytm stosuje opisane powyżej programowanie dynamiczne, lecz dla niektórych zbiorów X, uznaje, że nie opłaca się wykonywać zadań ze zbioru X jako pierwszych $|X|$ zadań w szukanym szeregowaniu, i przerywa stosowne obliczenie $M[X]$. Przy implementacji programowania dynamicznego jako rekurencyjnego obliczania wartości tablicy M z zastosowaniem spamiętywania, takie podejście daje algorytm o złożoności czasowej z grubsza równej liczbie nieodrzuconych zbiorów X (z dokładnością do czynnika wielomianowego w n). Decyzja, czy opłaca się obliczać wartość $M[X]$, jest oparta na wnikliwej analizie własności rozwiązania optymalnego.

3 Złożoność parametryzowana

Przytoczmy opowieść o doktorze $\mathcal{O}$, pochodzącą z książki Downeya i Fellowsa [12]. Doktor $\mathcal{O}$ przeprowadził eksperyment naukowy, zbierając wiele zestawów danych. W trakcie analizy danych okazało się, że niektóre pary zestawów są sprzeczne ze sobą. Doktor $\mathcal{O}$ podejrzewa, że niewielka część pomiarów została przeprowadzona nieprawidłowo, i należy je odrzucić (lub poddać głębszej analizie). Modelując zestawy danych jako wierzchołki grafu, a pary sprzecznych zestawów jako krawędzie, otrzymujemy znany problem

pokrycia wierzchołkowego: doktor $\mathcal{O}$ chce odrzucić jak najmniej zestawów danych (wierzchołków), które by usunęły wszystkie sprzeczności (krawędzie).

Problem pokrycia wierzchołkowego jest NP-zupełny, lecz doktor $\mathcal{O}$ nie jest na straconej pozycji: jego ratunkiem jest to, że liczba błędnych pomiarów — wielkość pokrycia wierzchołkowego — jest prawdopodobnie bardzo mała, powiedzmy rzędu 20, przy dużo większej liczbie zestawów danych. Prosty algorytm rozgałęziający rozwiązuje problem sprawdzania, czy graf o n wierzchołkach ma pokrycie wierzchołkowe rozmiaru k, w czasie $O(n2^k)$ — całkowicie akceptowalnym dla $k \sim 20$ i bardzo dużej wartości n. Warto porównać ten algorytm do podejścia całkowicie naiwnego, sprawdzającego wszystkie możliwe rozwiązania, działającego w czasie $O(n^k)$: dla $k = 20$, algorytm ten jest niepraktyczny właściwie dla każdej wartości n.

Problem doktora $\mathcal{O}$ jest jednym z wielu przykładów świadczących, że, dla parametru k o małej wartości, algorytmy o złożonośći $O(n^k)$ są niepraktyczne, podczas gdy algorytmy o złożoności $O(f(k)n^c)$ dla stałej c i funkcji obliczalnej (np. wykładniczej) f radzą sobie bardzo dobrze nawet z danymi o dużej wielkości (dużej wartości n). Algorytmy drugiego typu nazywamy algorytmami parametryzowanymi (ang. *fixed-parameter algorithms*), i w wielu sytuacjach ich opracowywanie jest motywowane praktycznymi zastosowaniami. Dodajmy, że w zastosowaniach często dane wejściowe są specyficzne: uchwycenie tej specyfiki w postaci odpowiedniego parametru pozwala na opracowywanie i precyzyjną analizę algorytmów, które w tych sytuacjach efektywnie rozwiązują problemy NP-zupełne.

Dziedzina złożoności parametryzowanej (ang. *parameterized complexity*), zajmująca się analizą algorytmów parametryzowanych, wraz z teorią dowodzącą, że w wielu przypadkach ich otrzymanie jest prawdopodobnie niemożliwe, została zainicjowana w latach 90. ubiegłego wieku przez Downeya i Fellowsa. Obecnie jest prężnie rozwijającą się dziedziną.

Bardzo pokrewnym pojęciem do algorytmów parametryzowanych jest kernelizacja. Algorytm kernelizacyjny to taki, który mając daną instancję I pewnego problemu, wraz z parametrem k, przekształca ją w równoważną instancję I' o wielkości ograniczonej przez $g(k)$ dla pewnego funkcji obliczalnej g. Prosta argumentacja pokazuje, że istnienie algorytmu kernelizacyjnego dla danego problemu rozstrzygalnego jest równoważne istnieniu algorytmu parametryzowanego dla tego problemu.

Podstawowym zadaniem kernelizacji jest wstępne przetworzenie danych wejściowych dla danego problemu, zanim poddamy je obróbce innym (wykładniczym, aproksymacyjnym) algorytmem. Z tego powodu najbardziej cenne są algorytmy kernelizacyjne, dla których funckja g — ograniczenie na wielkość powstającej instancji — jest wielomianem. W 2008 roku zo-

stała opracowana technika pokazywania, że dla danego problemu, przy odpowiednich założeniach teoriozłożonościowych, niemożliwe jest otrzymanie algorytmu kernelizacyjnego z wielomianowym ograniczeniem rozmiaru wyjścia.

W mojej rozprawie doktorskiej opisuję dwa wyniki z dziedziny złożoności parametryzowanej. Pierwszy z nich jest algorytmem parametryzowanym dla problemu Subset Feedback Vertex Set, parametryzowanego wielkością rozwiązania. Drugi, to dowód, że wiele problemów z wymogiem spójności nie ma algorytmu kernelizacyjnego z wielomianowym ograniczeniem na wielkość instancji wyjściowej, nawet przy założeniu, że wejściowy graf ma ograniczoną degenerację (ang. *degeneracy*).

3.1 Subset Feedback Vertex Set

W problemie Feedback Vertex Set (FVS) pytamy, czy z danego grafu nieskierowanego można usunąć k wierzchołków tak, by otrzymać las (graf acykliczny). Problem ten jest jednym z 21 problemów NP-zupełnych na liście Karpa i jest jednym z najgłębiej przestudiowanych problemów w dziedzinie złożoności parametryzowanej. Na uwagę zasługuje algorytm kernelizacyjny dla FVS autorstwa Thomassé [26], redukujący rozmiar instancji wejściowej do $O(k^2)$ wierzchołków i krawędzi, a korzystający z ciekawych twierdzeń z zakresu skojarzeń, takich jak twierdzenie Madera o S-ścieżkach [25].

W mojej rozprawie doktorskiej rozważam wariant problemu FVS zwany Subset Feedback Vertex Set (Subset-FVS), gdzie część wierzchołków wejściowego grafu jest wyróżnionych, a celem jest usunięcie k wierzchołków tak, by nie pozostał żaden cykl, który przechodzi przez co najmniej jeden wierzchołek wyróżniony.

Oczywiście, problem Subset-FVS jest co najmniej tak samo trudny jak FVS. Okazuje się też, że Subset-FVS uogólnia też problem Multiway Cut, w którym, mając dany graf z wyróżnionymi terminalami, należy usunąć k wierzchołków tak, by każdy terminal pozostał w innej spójnej składowej. Problemy rozcinania grafu, z których najprostszym jest Multiway Cut, są obecnie jednym z najintensywniej analizowanych zagadnień w złożoności parametryzowanej, a ich badanie doprowadziło do odkrycia techniki ważnych cięć (ang. *important separators*) [22] i kernelizacji przez zanurzanie w matroidach [21]. Badanie problemu Subset-FVS wpisuje się więc w ten nurt.

W 2009 roku pytanie o istnienie algorytmu parametryzowanego dla problemu Subset-FVS postawili niezależnie Kawarabayashi i Saurabh. W rozprawie doktorskiej odpowiadam na to pytanie twierdząco:

Twierdzenie 4. *Problem* Subset Feedback Vertex Set *można rozwiązać w czasie* $O^\star(c^{k \log k})$*, gdzie* c *jest pewną stałą, a* k *jest liczbą wierzchołków do usunięcia.*

Opracowany algorytm dla problemu Subset-FVS jest dość złożony i wielostopniowy. Proste przekształcenie pokazuje, że w problemie Subset-FVS możemy rozważać graf z wyróżnionymi krawędziami, a nie wierzchołkami. Najpierw dowodzimy, że jeśli liczba wyróżnionych krawędzi jest mała (ograniczona przez funkcję k), to problem można zredukować do problemu rozwiązywania wielu egzemplarzy problemu Multiway Cut. Następnie podajemy zestaw reguł redukcyjnych i rozgałęziających, które pozwalają zredukować liczbę wyróżnionych krawędzi do $O(k^3)$. Nasze reguły korzystają m.in. z narzędzi opracowanych przez Thomassé [26], z techniki iterative compression [24] oraz z kilku pomysłów stosowanych przy jednym ze starszych algorytmów dla FVS autorstwa Guo i innych [19].

3.2 Trudność kernelizacji w grafach o ograniczonej degeneracji

W 2004 roku ukazała się przełomowa praca Alber i innych [1] pokazująca, że problem znajdowania minimalnego zbioru dominującego parametryzowanego wielkością rozwiązania k, dla którego prawdopodobnie nie istnieje algorytm parametryzowany w ogólności, ma algorytm kernelizacyjny redukujący dane do rozmiaru $O(k)$ w przypadku, gdy graf wejściowy jest planarny. Opracowana w tej pracy technika podziału na regiony okazała się mieć dużo szersze zastosowania, i dalsze badania zostały zwieńczone odryciem algorytmów kernelizacyjnych dla wielu problemów w przypadku, gdy graf wejściowy należy do pewnych klas grafów rzadkich, takich jak grafy o ograniczonym genusie lub z wykluczonym ustalonym minorem [17].

Powyższe wyniki bardzo mocno wykorzystują topologiczną strukturę grafów z wykluczonym ustalonym minorem. Inną klasą grafów rzadkich, nie mającą żadnych topologicznych własności, jest klasa grafów d-zdegenerowanych: graf jest d-zdegenerowany, jeśli każdy jego podgraf ma wierzchołek o stopniu co najwyżej d. Przykładowo, lasy to klasa grafów 1-zdegenerowanych, a każdy graf planarny jest 5-zdegenerowany. Można pokazać, że graf o wykluczonym ustalonym minorze jest $O(1)$-zdegenerowany, gdzie stała degeneracji zależy od wykluczonego minora. Tak więc klasy grafów zdegenerowanych posiadają podobne własności co do stopni wierzchołków, co klasy grafów o wykluczonym ustalonym minorze, lecz nie posiadają własności topologicznych (co można zobaczyć obserwując, że wkładając wierzchołek w każdą krawędź dowolnego grafu otrzymujemy graf 2-zdegenerowany).

Okazuje się, że założenie, że graf wejściowy jest d-zdegenerowany dla ustalonej stałej d pozwala opracować algorytmy parametryzowane i kernelizacyjne w wielu przypadkach, w których w ogólności takie wyniki prawdopodobnie nie istnieją. Najgłośniejszym taki wynikiem jest opracowanie przez Philipa i innych [23] algorytmu kernelizacyjnego o wielomianowym ograniczeniu na rozmiar wyjścia, dla problemu minimalnego zbioru dominującego w grafach d-zdegenerowanych.

Na pierwszych warsztatach o kernelizacji (WorKer 2009) zostało postawione pytanie, czy ograniczona degeneracja grafu pomaga w kernelizacji innych problemów, w tym problemów z wymogiem spójności. W moje rozprawie odpowiadam na to pytanie przecząco.

Twierdzenie 5. *Jeśli NP $\not\subseteq$ coNP/poly, dla dowolnego $d \geq 2$, nie istnieją algorytmy kernelizacyjne redukujące rozmiar instancji do rozmiaru zależnego wielomianowo od parametru, dla wielu problemów z wymogiem spójności, w tym* Connected Dominating Set*, parametryzowanych rozmiarem rozwiązania, nawet przy założeniu, że wejściowy graf jest d-zdegenerowany.*

Warto zwrócić uwagę, że NP $\subseteq$ coNP/poly powoduje ustalenie się hierarchii wielomianowej na trzecim poziomie [6, 28].

Dowód powyższego twierdzenia używa narzędzi dowodzenia trudności kernelizacji opracowanych w 2008 roku przez Fortnowa i Santhanama [18] oraz Bodlaendera i innych [5]. Kluczowym pomysłem jest użycie właściwego dodatkowego problemu pośredniego w redukcjach — problemu Graph Motif — który dobrze abstrahuje wymóg spójności. W problemie tym mamy dany graf nieskierowany $G = (V, E)$ oraz funkcję (kolorowanie) $f : V \to \{1, 2, \ldots, k\}$ i pytamy o zbiór $C \subseteq V$ składający się z k wierzchołków taki, że C indukuje spójny podgraf G oraz f jest bijekcją na C (tj., C składa się z jednego wierzchołka każdego koloru). Problem Graph Motif jest NP-zupełny nawet w przypadku drzew o maksymalnym stopniu 3 [13]. Przy parametryzacji liczbą kolorów (k) łatwo pokazać korzystając w technik Bodlaendera i innych, że prawdopodobnie nie posiada on algorytmu kernelizacyjnego redukującego liczbę wierzchołków do zależnej wielomianowo od k nawet w przypadku, gdy graf wejściowy jest lasem. Problem Graph Motif jest też wystarczająco prosty, by pozwolić na proste redukcje do rozważanych problemów w grafach 2-zdegenerowanych.

4 Uwagi końcowe

Nowe wyniki zawarte w rozprawie zostały uzyskane razem z moimi współpracownikami: Markiem Cyganem, Michałem Pilipczukiem i Jakubem Onufrym

Wojtaszczykiem, i zostały opublikowane w następujących miejscach:

1. Algorytm rozwiązujący problem Capacitated Dominating Set był przedstawiony na konferencji SWAT 2010 [10], zaś sam artykuł został przyjędy do czasopisma Information Processing Letters.

2. Algorytm rozwiązujący problem Minimum Maximal Irredundant Set był przedstawiony na konferencji CIAC 2010 [11], zaś sam artykuł został przyjęty do czasopisma Journal of Discrete Algorithms [2].

3. Algorytm rozwiązujący problem $1|\text{prec}|\sum C_i$ był przedstawiony na konferencji ESA 2011 [8].

4. Algorytm parametryzowany dla problemu Subset Feedback Vertex Set był przedstawiony na konferencji ICALP 2011 [9].

5. Dowody trudności kernelizacji problemów w grafach o ograniczonej degeneracji były przedstawione na konferencji WG 2010 [7].

Literatura

[1] J. Alber, M. R. Fellows, R. Niedermeier. Polynomial-time data reduction for dominating set. *J. ACM*, 51(3):363–384, 2004.

[2] D. Binkele-Raible, L. Brankovic, M. Cygan, H. Fernau, J. Kneis, D. Kratsch, A. Langer, M. Liedloff, M. Pilipczuk, P. Rossmanith, J. O. Wojtaszczyk. Breaking the 2^n-barrier for irredundance: Two lines of attack. *J. Discrete Algorithms*, 9(3):214–230, 2011.

[3] A. Björklund. Determinant sums for undirected hamiltonicity. *FOCS*, strony 173–182. IEEE Computer Society, 2010.

[4] A. Björklund, T. Husfeldt, P. Kaski, M. Koivisto. Fourier meets möbius: fast subset convolution. D. S. Johnson, U. Feige, redaktorzy, *STOC*, strony 67–74. ACM, 2007.

[5] H. L. Bodlaender, R. G. Downey, M. R. Fellows, D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[6] J. Cai, V. T. Chakaravarthy, L. A. Hemaspaandra, M. Ogihara. Competing provers yield improved Karp-Lipton collapse results. *Inf. Comput.*, 198(1):1–23, 2005.

[7] M. Cygan, M. Pilipczuk, M. Pilipczuk, J. O. Wojtaszczyk. Kernelization hardness of connectivity problems in d-degenerate graphs. D. M. Thilikos, redaktor, *WG*, wolumen 6410 serii *Lecture Notes in Computer Science*, strony 147–158, 2010.

[8] M. Cygan, M. Pilipczuk, M. Pilipczuk, J. O. Wojtaszczyk. Scheduling partially ordered jobs faster than 2^n. C. Demetrescu, M. M. Halldórsson, redaktorzy,

ESA, wolumen 6942 serii *Lecture Notes in Computer Science*, strony 299–310. Springer, 2011.

[9] M. Cygan, M. Pilipczuk, M. Pilipczuk, J. O. Wojtaszczyk. Subset feedback vertex set is fixed-parameter tractable. L. Aceto, M. Henzinger, J. Sgall, redaktorzy, *ICALP (1)*, wolumen 6755 serii *Lecture Notes in Computer Science*, strony 449–461. Springer, 2011.

[10] M. Cygan, M. Pilipczuk, J. O. Wojtaszczyk. Capacitated domination faster than $O(2^n)$. H. Kaplan, redaktor, *SWAT*, wolumen 6139 serii *Lecture Notes in Computer Science*, strony 74–80. Springer, 2010.

[11] M. Cygan, M. Pilipczuk, J. O. Wojtaszczyk. Irredundant set faster than $\mathrm{O}(2^n)$. T. Calamoneri, J. Díaz, redaktorzy, *CIAC*, wolumen 6078 serii *Lecture Notes in Computer Science*, strony 288–298. Springer, 2010.

[12] R. G. Downey, M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[13] M. R. Fellows, G. Fertin, D. Hermelin, S. Vialette. Sharp tractability borderlines for finding connected motifs in vertex-colored graphs. L. Arge, C. Cachin, T. Jurdzinski, A. Tarlecki, redaktorzy, *ICALP*, wolumen 4596 serii *Lecture Notes in Computer Science*, strony 340–351. Springer, 2007.

[14] F. Fomin, K. Iwama, D. Kratsch. Moderately exponential time algorithms, dagstuhl seminar, 2008.

[15] F. V. Fomin, F. Grandoni, D. Kratsch. A measure & conquer approach for the analysis of exact algorithms. *J. ACM*, 56(5), 2009.

[16] F. V. Fomin, D. Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, wydanie 1, 2010.

[17] F. V. Fomin, D. Lokshtanov, S. Saurabh, D. M. Thilikos. Bidimensionality and kernels. M. Charikar, redaktor, *SODA*, strony 503–510. SIAM, 2010.

[18] L. Fortnow, R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. *J. Comput. Syst. Sci.*, 77(1):91–106, 2011.

[19] J. Guo, J. Gramm, F. Hüffner, R. Niedermeier, S. Wernicke. Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. *J. Comput. Syst. Sci.*, 72(8):1386–1396, 2006.

[20] M. Held, R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of SIAM*, 10:196–210, 1962.

[21] S. Kratsch, M. Wahlström. Compression via matroids: A randomized polynomial kernel for odd cycle transversal. *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, strona to appear, 2012.

[22] D. Marx. Parameterized graph separation problems. *Theor. Comput. Sci.*, 351(3):394–406, 2006.

[23] G. Philip, V. Raman, S. Sikdar. Solving dominating set in larger classes of graphs: FPT algorithms and polynomial kernels. A. Fiat, P. Sanders, redaktorzy, *ESA*, wolumen 5757 serii *Lecture Notes in Computer Science*, strony 694–705. Springer, 2009.

[24] B. A. Reed, K. Smith, A. Vetta. Finding odd cycle transversals. *Oper. Res.*

Lett., 32(4):299–301, 2004.

[25] A. Schrijver. A short proof of Mader's sigma-paths theorem. *J. Comb. Theory, Ser. B*, 82(2):319–321, 2001.

[26] S. Thomassé. A $4k^2$ kernel for feedback vertex set. *ACM Transactions on Algorithms*, 6(2), 2010.

[27] G. J. Woeginger. Space and time complexity of exact algorithms: Some open problems (invited talk). R. G. Downey, M. R. Fellows, F. K. H. A. Dehne, redaktorzy, *IWPEC*, wolumen 3162 serii *Lecture Notes in Computer Science*, strony 281–290. Springer, 2004.

[28] C.-K. Yap. Some consequences of non-uniform conditions on uniform classes. *Theor. Comput. Sci.*, 26:287–300, 1983.