Uniwersytet im. A. Mickiewicza w Poznaniu

Wydział Matematyki i Informatyki

kierunek: Informatyka

Praca magisterska

Analiza algorytmów dla wybranych problemów NP-zupełnych

Kamil Kostyszyn

Promotor:

prof. dr. hab. Zbigniew Palka

Poznań, 2014

Poznań, ...

Oświadczenie

Ja, niżej podpisany/a .....
student/ka Wydziału Matematyki i Informatyki Uniwersytetu im. Adama Mickiewicza w Poznaniu oświadczam, że przedkładaną pracę dyplomową pt:
.....
napisałem/am samodzielnie. Oznacza to, że przy pisaniu pracy, poza niezbędnymi konsultacjami, nie korzystałem/am z pomocy innych osób, a w szczególnosci nie zlecałem/am opracowania rozprawy lub jej cześci innym osobom, ani nie odpisywałem tej rozprawy lub jej cześci od innych osób. Oświadczam również , że egzemplarz pracy dyplomowej w formie wydruku komputerowego jest zgodny z egzemplarzem pracy dyplomowej w formie elektronicznej. Jednocześnie przyjmuję do wiadomości, że gdyby powyższe oświadczenie okazało sie nieprawdziwe, decyzja o wydaniu mi dyplomu zostanie cofnięta.

.....

Spis treści

Wstęp

Zasadniczym celem pracy dyplomowej przygotowywanej przez studenta jest:

- wykazanie sie umiejętnością formułowania i rozwiązywania problemów wiążących się z programem odbytych studiów,
- wykazanie sie znajomością metod i sposobów prowadzenia analizy oraz redakcyjnego przygotowania pracy w oparciu o umiejętności nabyte w czasie studiów.

Praca dyplomowa musi być samodzielnym opracowaniem autorstwa studenta, przygotowanym przy pomocy promotora. Student, jako autor ponosi pełna odpowiedzialność z tytułu oryginalności i rzetelności zaprezentowanego materiału i powinien uwzględniać wszelkie prawa i dobre obyczaje w tym zakresie.

Tematyka pracy powinna znacząco wykraczać poza materiał omówiony w trakcie studiów; zakłada samodzielny wkład autora w postaci np.: implementacji, opracowania algorytmu, samodzielnego porównania, oceny i analizy istniejących rozwiązań.

Rozdział 1

Wprowadzenie

Aby dobrze przedstawić problemy i algorytmy będące tematem pracy, niezbędne jest przytoczenie podstawowych pojęć oraz zagadnień odnoszących się NP-zupełności.

Podstawowym pojęciem, które w pierwszej kolejności należało by przedstawić jest samo pojęcie "problemu". Według definicji w [1] problem abstrakcyjny Q jest relacją dwuargumentową, określoną na zbiorze I egzemplarzy problemu oraz zbiorze S rozwiązań problemu. Egzemplarzem jest struktura matematyczna, natomiast podzbiór jej elementów składowych jest rozwiązaniem. Przenosząc to na przykład dla problemu MAXIMUM-INDEPENDET-EDGE-SET (czyli znalezienia maksymalnego skojarzenia) egzemplarzem jest graf G złożony ze zbioru wierzchołków V oraz krawędzi E. Zatem maksymalny zbiór krawędzi nie mających wspólnych wierzchołków jest rozwiązaniem tego problemu. Skoro problem jest relacją, a nie funkcją możliwa jest sytuacja, że dla jednego egzemplarza może istnieć wiele rozwiązań.

W przypadku poruszania się w obrębie teorii NP-zupełności takie definiowanie problemu jest zbyt ogólne dlatego najlepiej jest ograniczyć się do pro-

blemów decyzyjnych. W tej formie znana jest instancja problemu oraz wynik będący odpowiedzią. W przypadku INDEPENDET-EDGE-SET egzemplarz ma postać $i =< G, k >$, a rozumiemy go: w grafie G istnieje skojarzenie o mocy co najmniej k. Rozwiązaniem jest podzbiór zbioru krawędzi $E' \in E$ o mocy co najmniej k. Zadnie polega na zweryfikowaniu czy podany wynik jest rzeczywiście prawdziwy. Nie trudno zauważyć że dla tak postawionego problemu zbiór odpowiedzi w takim przypadku ogranicza się do dwóch wartości prawdy lub fałszu. Pozornie łatwe podejście w celu rozpatrywania problemu nie jest wcale takie proste gdy dochodzi do jego realizacji.

Inną formą podejścia do problemu jest podejście optymalizacyjne. Podobnie jak w problemie decyzyjnym na wstępie otrzymujemy instancje problemu, jednak nie występuje sugerowane rozwiązanie. Naszym zadaniem jest teraz znalezienie najlepszego wyniku, w zależności czy nasz zbiór wynikowy chcemy minimalizować (w celu zmniejszania pewnych kosztów) czy też zależy nam na maksymalizowaniu (w celu zwiększenia pewnych zysków). Odpowiedzą tak sformułowanemu problemowi będzie moc zbioru wynikowego, czasami również elementy tego zbioru. Mimo iż optymalizacyjne podejście wydaję się przy realizacji trudniejsze, można je łatwo zastąpić formą decyzyjną w bardzo prosty sposób. Należy obrać początkową trywialną wartość, a następnie zmieniać ją o jednostkową wartość do momentu aż przestanie istnieć rozwiązanie dla naszej wartości. W przypadku minimalizacji za trywialną wartość obieramy maksymalną wartość dla zbioru, którą później dekrementujemy. Natomiast dla maksymalizacji trywialną wartością startową jest zero, które zwiększamy aż do momentu w którym przestanie istnieć rozwiązanie.

Większa część problemów należy do klasy złożoności P, dlatego rzadko podejmuje się problemy wybiegające poza tę klasę. Problemy będące tematem tej pracy wybiegają dalej i dotyczą sławnej hipotezy: "czy P $\neq$ NP",

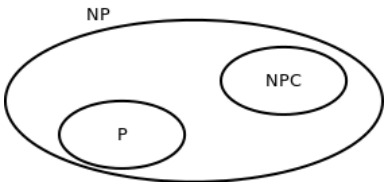
dlatego obszar który podejmuje praca odnosi się do klas problemów P, NP i NPC. Praca zakłada że P $\neq$ NP w innym przypadku wywód ten nie ma sensu.

Najbardziej powszechna klasa P zawiera ten zbiór problemów dla których istnieje algorytm znajdujący rozwiązanie w czasie wielomianowym. Oznaczymy czas działania algorytmu jako funkcje f która będzie działać ze zbioru oznaczającego rozmiar danych o ustalonym kodowaniu na zbiór odpowiadający liczbie operacji elementarnych potrzebnych do zakończenia działania algorytmu. Przy tak określonej funkcji jeśli możemy ją ograniczyć z góry przez dowolną funkcję wielomianową $O(nk)$ to wtedy problem należy do klasy złożoności P.

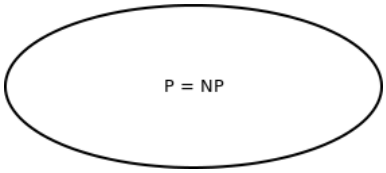
O ile P $\neq$ NP to klasa NP zawiera te problemy które są "weryfikowalne" w czasie wielomianowym. Przez słowo "weryfikowalne" rozumiemy, że istnieje takie świadectwo pozytywne (zwane inaczej dowodem), dzięki któremu jesteśmy w stanie sprawdzić czy otrzymane rozwiązanie jest poprawne. Jak widać klasa P $\subseteq$ NP z tej racji iż jeśli możemy rozwiązać problem w czasie wielomianowym to również jesteśmy w stanie w wielomianowym czasie sprawdzić pozytywność świadectwa.

Klasa NPC jest tym co różni klasę P i NP, w jej skład wchodzą te problemy które należą do NP a nie należących do P. Nasuwa się pytanie: "Czy w ogóle istnieją takie problemy z klasy NP, których nie można rozwiązać w czasie wielomianowym?". Na obecny stan rzeczy odpowiedz brzmi: nie wiadomo. W czasie powstawania pracy nikt dotąd nie udowodnił że wszystkie problemy $Q \in$ NP są rozwiązywalne w czasie wielomianowym, ani też nie pokazał że istniej taki problem z klasy NPC który nie jest rozwiązany w czasie wielomianowym. Tematem tej pracy właśnie są te problemy o których wiem że są weryfikowalne w czasie wielomianowym, a nie są rozwiązywalne

w czasie wielomianowym.



Rys. 1.1. P $\neq$ NP $\wedge$ NPC $\neq \emptyset$



Rys. 1.2. P = NP $\wedge$ NPC $= \emptyset$

Szczególną własnością problemów NPC jest że jeśli choć jeden problem znajdzie się w klasie P to wszystkie problemy z klasy NPC będą należały do klasy P. Pokazuje to że jeśli komuś uda się znaleźć algorytm wielomianowy rozwiązujący jakikolwiek problem z klasy NPC to oznaczało by że P = NP. Związanie ze sobą tych wszystkich problemów NPC zawdzięczamy redukcji wielomianowej, polega ona na tym że rozwiązanie pewnego problemu można sprowadzić do rozwiązania innego. Odbywa się to za pośrednictwem algorytmu redukcji, który jest w stanie przeprowadzić każdą instancję problemu A do postaci instancji problemu B w czasie wielomianowym. Aby to zobrazować posłużę się trywialnym przykładem w którym problem Q obliczenia długości wektora w $\mathbb{R}^2$ zredukujemy do problemu Q' obliczenia długości wektora w $\mathbb{R}^3$. Mając tak zdefiniowane problemy potrzebujemy jeszcze algorytm redukcji. Nasz algorytm będzie rozszerzał wektor o trzecią pozycję i wpisywał w to miejsce 0. Teraz każdy wektor z $\mathbb{R}^2$ możemy przedstawić w $\mathbb{R}^3$ a następnie obliczyć jego długość. Widać że długości wektora $[x1, x2, 0]$ odpowiada długości wektora $[x1, x2]$. Pokazuje to że problem Q można zredukować do problemu Q'.

Mimo iż tak wiele wiadomo o problemach NP-zupełnych to nadal sprawiają one wyzwanie. O ile algorytmy o złożoności większej od wielomianowej potrafią znaleźć optymalne rozwiązanie, o tyle należy mieć pewność, że dane będą z odpowiednio wąskiego zakresu. Jeśli jednak zakres danych jest zbyt

duży pozostaje jedynie heurystyczne poszukiwanie wyniku dla mocno sprecyzowanego problemu lub zadowolenie się wynikiem zbliżonym do optymalnego w oparciu o algorytm aproksymacyjny. W przypadku heurystycznego podejścia nie można za wiele powiedzieć gdyż wszystko zależy od indywidualnej potrzeby o którą musi zadbać algorytm oraz improwizacji na którą można sobie pozwolić.

Obecnie najbardziej popularną metodą jest wykorzystanie algorytmów aproksymacyjnych ze względu na ich wielomianową złożoność. Jakość takiego algorytmu zależy od znalezionego rozwiązania C względem rozwiązania optymalnego C^*. Współczynnik aproksymacji $1 \leq \rho(n)$ jest wyznacznikiem jakości i ogranicza z góry stosunek kosztów: $max(\frac{C}{C^*}, \frac{C^*}{C}) \leq \rho(n)$

Wzór ten uwzględnia przypadek maksymalizacji jak i minimalizacji określając ile razy otrzymane rozwiązanie jest gorsze od optymalnego. Dla maksymalizacji mamy $0 \leq C \leq C^*$ czyli wartość znalezionego rozwiązania jest nie większa od rozwiązania optymalnego. Natomiast w problemie minimalizacji $0 \leq C^* \leq C$ wartość znalezionego rozwiązania jest nie mniejsza od rozwiązania optymalnego.

Rozdział 2

Pierwszy problem NP

2.1. Wstęp

Aby przejść do rozważań nad konkretnymi problemami z klasy NP-zupełnej należy wpierw skupić się nad znaczeniem tej klasy. W teori NP-zupełności fundamentalne znaczenie ma metoda redukcji pomiędzy dwoma problemami. Podstawą wykorzystania tej metody jest założenie że klasa ta jest niepusta dzięki czemu możemy pokazać redukcję problemu, który chcemy wprowadzić do tej klasy względem problemu znajdującego się już w klasie NP-zupełnej. Jak widać początki zawsze są najtrudniejsze z tej racji "jak tego dokonać jeśli w początkowych rozważaniach klasa NPC jest pusta?". Oczywiście nie można posłużyć się redukowalnością, aby wprowadzić pierwszy "pierwotny" problem. Niezbędne okaże się skorzystanie wprost z definicji, by w ten oto sposób zasilić klasę NP-zupełną o pierwszy problem. Później w oparciu o to będziemy mogli już redukcją wprowadzać pozostałe problemy.

2.2. Charakterystyka klasy NP-zupełnej

Definicja NP-zupełności mówi że, aby problem należał do klasy NPC musi spełniać dwa warunki, które można sprowadzić do następujących własności:

1. świadectwo problemu jest weryfikowalne w czasie wielomianowym
2. jest co najmniej tak "trudny", jak każdy problem z NP.

Własność 1. nie wydaje się względnie trudnym warunkiem, natomiast warunek 2., choć wygląda niepozornie może przysporzyć wielu kłopotów. Własność ta mówi o tym, że każdy problem z klasy NP można zredukować do naszego problemu. Tu zaczynają się komplikacje, ponieważ jesteśmy zmuszeni do określenia metody redukcji każdego problemu z klasy NP (nie ważne jaki on by nie był) do naszego problemu "pierwotnego". Choć pozornie to zadanie wydaje się nie do zrealizowania o tyle istnieje na to bardzo ciekawe rozwiązanie. Błyskotliwe spostrzeżenie będące rozwiązaniem naszej zagadki jest ukryte w komputerach, a dokładnie na relacji pomiędzy modelem danych, a jego niższą warstwą fizycznych danych. Oczywiste jest to, że algorytmy implementujemy w pamięci komputera, zaś ten jest w stanie przeprowadzić jego działanie, tak aby otrzymać wynik. Natomiast cała idea tego przetwarzania jest realizowania na gigantycznych sieciach logicznych. Przyjrzyjmy się zatem problemowi spełnialności układów logicznych jako potencjalnemu kandydatowi na rolę "pierwotnego" problemu w klasie NPC.

2.3. Problem spełnialności układów logicznych

Problem spełnialności układów logicznych (CIRCUIT-SAT) to układ C zbudowany ze stałej liczby wejść binarnych oraz bramek logicznych. W pod-

stawowym charakterze wyróżniamy bramki AND, OR I NOT. Układ nazywamy spełnialnym jeśli istnieje takie wartościowanie wejść sprawiające iż po wykonaniu wszystkich operacji logicznych z bramek na wyjściu otrzymamy prawdę.

Chcąc znaleźć rozwiązanie tego problemu istniej jedynie najbardziej prymitywny sposób jego realizacji. Polega on na sprawdzeniu wszystkich możliwych odpowiedzi, a dokładniej w obecnym tu przypadku na zbadaniu wszystkich możliwych wartościowań wejścia układu. Zakładając że układ składa się z k wejść od razu widać że należy sprawdzić 2^k możliwych przypadków niezależnie od rozmiaru egzemplarza danych wejściowych. Sprawia to, że czas działania takiego algorytmu jest wykładniczy, czyli wyższy od wielomianowego.

Spróbujmy wykazać teraz, że CIRCUIT-SAT $\in$ NPC. W tym celu wpierw skupmy się na pierwszej własności i udowodnijmy, że pozytywne świadectwo można weryfikować w czasie wielomianowym, czyli w rezultacie CIRCUIT-SAT $\in$ NP.

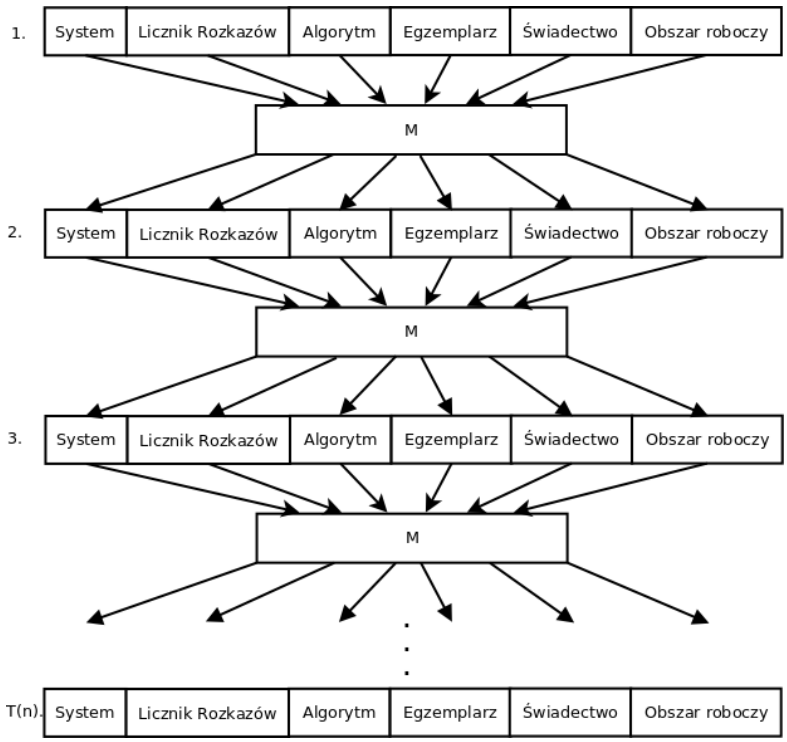
Lemat 1. *CIRCUIT-SAT* $\in$ *NP*

Dowód: Algorytm weryfikujący problem CIRCUIT-SAT na wejściu oczekuje dwóch parametrów. Pierwszy to egzemptlarz E pełnej specyfikacji układu logicznego C, natomiast drugi parametr to świadectwo y zawierające binarne wartości przypisane do konkretnych wejść. Pierwszym krokiem algorytmu jest sprowadzenie układu logicznego C do zapisu w formie odwrotnej notacji polskiej. W tym procesie bramki logiczne zostają przekształcone na operandy logiczne w następujący sposób: AND jako koniunkcja, OR jako alternatywa, a NOT negacja. Przy rozsądnej specyfikacji układu logicznego krok ten można wykonać w sposób liniowy. Następną czynnością jest wykonanie tych operacji względem wartości logicznych z świadectwa y. Do uzyska-

nia poprawnej kolejności wykonywanych operacji na bitach wykorzystuje się strukturę kolejki i stosu. Po zrealizowaniu wszystkich operacji logicznych na końcu stosu znajduje się wynik będący jedynym elementem stosu. Rozwiązanie jest poprawne, gdyż zmianie realizacji problemu uległa jedynie forma z bramek logicznych na operacje logiczne, a wynik jest jedynie konsekwencją realizacji działań logicznych. Jeśli w wyniku na końcu uzyskaliśmy prawdę (wartość logiczną 1) oznacza to że rzeczywiście świadectwo y spełnia układ logiczny C z egzemplarza E. W przeciwnym przypadku, kiedy wynikiem był by fałsz dostajemy informacje, że to świadectwo nie świadczy o spełnialności układu logicznego C, zarazem nie wykluczając jego spełnialności. W przypadku gdy układ C nie był by spełnialny to nie ważne dla jakiego wartościowania nigdy nie uzyskamy prawdy na końcu przy poprawnym wykonaniu wszystkich operacji logicznych. W kwestii złożoności czasowej, łatwo zauważyć, że algorytm składa się z dwóch kroków, z czego każdy ma złożoność liniową. Sprawia to, że całość algorytmu realizowana jest w czasie liniowym, a zatem i wielomianowym, co kończy dowód i ukazuje, że CIRCUIT-SAT $\in$ NP.

Po wykazaniu iż CIRCUIT-SAT $\in$ NP następnym krokiem jest pokazanie, że CIRCUIT-SAT $\in$ NP-trudny, czyli w praktyce odpowiada temu realizacja własności 2. Przebieg tego dowodu będzie wykorzystywał ściśle zasadę funkcjonowania komputera. Naszą rolą będzie sprawienie by problem CIRCUIT-SAT realizował schemat obliczeniowy dowolnego algorytmu rozwiązującego problem z klasy NP na komputerze. Nim do tego przystąpimy należy bliżej nakreślić sytuacje w której realizowany będzie dowód. W pierwszej kolejności zwróćmy uwagę na fakt, że komputer przede wszystkim wykorzystuje pamięć i procesor. Pamięć służy do przechowywania wartości. Natomiast procesor składa się z maszyny, którą oznaczmy jako M zawierającej sieć bramek logicznych dzięki, której jest w stanie wykonywać operacje logiczne, a

co dalej za tym idzie wykonywać instrukcje. Skupmy się teraz na pamięci i podzielmy ją na obszary ze względu na naszą przydatność do realizacji algorytmu. Zdecydowanie największym obszarem, który zajmuje pamięć są dane związane z obsługą komputera umożliwiające jego działanie. Dodatkowo wyznaczmy obszar licznika rozkazów za pośrednictwem, którego znajduje się adres następnej komendy do wykonania. Teraz zajmijmy się obszarami, które bezpośrednio nas interesują, jest to obszar pamięci w której jest zapisany algorytm A, obszar na egzemplarz problemu x oraz świadectwo y. W praktyce jeszcze potrzebujemy jednego obszaru, którego przeznaczeniem będzie przechowywanie obliczeń częściowych, które będzie wykorzystywał algorytm A oraz po wykonaniu wszystkich operacji w pewnej ustalonej komórce tego obszaru będzie znajdował się nasz wynik. Konfiguracją komputera nazwijmy wszystkie te obszary. Działanie komputera należy opisać w następujący sposób: najpierw działa w sposób wcześniej zaprezentowany, następnie wykonuje obliczenia w oparciu o konfiguracje, a rezultat obliczeń zostaje zapisany jako nowa konfiguracja. W rezultacie jedna konfiguracja zostaje przekształcona w kolejną i tak bez końca, dopóki urządzenie funkcjonuje.



Rys. 2.1. Sekwencja konfiguracji

Zajmijmy się teraz sprawdzeniem czy problem spełnialności układów logicznych jest NP-trudny:

Lemat 2. *CIRCUIT-SAT* $\in$ *NP-trudny.*

Dowód: Aby wykazać, iż CIRCUIT-SAT jest NP-trudny pokażemy, że każdy problem z klasy NP można zredukować do CIRCUIT-SAT. W tym celu wyznaczymy kolejne etapy, które to przedstawią:

1. Zdefiniowanie algorytmu redukcji.

   Dla każdego problemu Q z klasy NP jesteśmy w stanie określić dwuparametrowy algorytm A, którego zadaniem jest zweryfikowanie świadectwa. W tym przypadku wykażemy zależność, iż jeśli świadectwo jest poprawne to można je przekształcić na odpowiednią postać binarną,

która z kolei jest spełniającym wartościowaniem układu logicznego C. Również w drugą stronę, jeśli znajdziemy takie wartościowanie, które spełnia układ logiczny C to będzie go można przeformułować na postać świadectwa które będzie poprawnie weryfikowane przez algorytm A.

Skupmy się teraz na konstrukcji która zamienia proces wykonywania algorytmu A na tworzenie układu logicznego C. Jak wcześniej wspomniałem, wszystkie dane zawierające informacje na temat komputera i algorytmu A znajdują się w pamięci komputera i wspólnie stanowią pierwszą konfiguracje. Następnie maszyna M przeprowadza kolejne kroki algorytmu, przechodząc z jednej konfiguracji w drugą, aż do momentu zakończenia działania algorytmu A. W rezultacie liczba konfiguracji jest dokładnie równa $T(n)$, czyli liczbie kroków jakie potrzebuje algorytm A do uzyskania wyniku końcowego. Jak widać przez cały ten czas maszyna M pozostaje niezmienna a jej postępowanie zależy jedynie od konfiguracji i przez ten czas jak będzie się zmieniała konfiguracja tak również maszyna M będzie zmieniała różne obszary pamięci.

Aby sprowadzić zasadę działania wykonywania algorytmu A w komputerze do struktury problemu CIRCUIT-SAT, to konfiguracja c_0 jest przewodami wejściowymi do układu logicznego. Następnie działa na to struktura logiczna maszyny M. W wyniku takiego działania otrzymujemy wyjście, które jest zapisywane jako konfiguracja c_1, jednak w naszym schemacie nie możemy zastosować takie rozwiązania w wyniku czego po prostu przedłużamy otrzymane ścieżki tak aby były kolejnym wejściem do powielonej maszyny M. Kontynuując to w ten sposób otrzymujemy tyle obszarów bramek logicznych ile wynosi liczba kroków algorytmu A. W końcowej fazie musimy wypuścić na wyjście ścieżkę z ostatniej konfiguracji $c_{T(n)}$ obszaru pamięci przeznaczonej na wynik al-

gorytmu. Jest to wyjście naszego układu zawierającego odpowiedz na pytanie czy świadectwo jest prawdziwe. W ten oto sposób niejako "skleiliśmy" ze sobą przejścia przez wszystkie konfiguracje,tak by na końcu całego układu znalazła się końcowa konfiguracja $c_T(n)$.

2. Analiza poprawności

   W ten sposób skonstruowany układ C, jak można zauważyć, jest spełnialny tylko wtedy gdy świadectwo jest pozytywnie zweryfikowane przez algorytm A, a nie może być inaczej gdyż układ C symuluje obliczeniową naturę komputera. Z drugiej strony jeśli na wejście układu C przedstawimy pewne wartościowanie które powoduje spełnialność układu oznacza, to że ta interpretacja binarna danych przekłada się na poprawne świadectwo, gdyż algorytm A poprawnie weryfikuje to świadectwo, a jak wiemy A jest poprawnym algorytmem. Zatem algorytm redukcji F jest skonstruowany w sposób poprawny.

3. Analiza złożoności

   Ostatnim elementem dowodu jest wykazanie wielomianowej złożoności algorytmu redukcyjnego F względem rozmiaru danych wejściowych $n = |x|$. Niewątpliwie rozmiar potrzebny na dane związane ze stanem maszyny, obszarem licznika rozkazów i implementacją maszyny M jest zależny od typu komputera który symuluje. Mimo to jest stały za względu na rozmiar danych przychodzących. Podobnie rozmiar algorytmu A jest wartością stałą. Kolejnym czynnikiem są dane wejściowe egzemplarza x oraz świadectwa y. Świadectwo będące podzbiorem egzemplarza jest nie większe rozmiarem od egzemplarza x, zaś rozmiar egzemplarza wynosi n. Ostatnim elementem przekładającym się na strukturze algorytmu F jest obszar roboczy w którym znajdzie się wynik oraz któ-

ry służy na pomocnicze obliczenia algorytmu złożoności wielomianowej A, czyli nie może być większy niż wielomianowa od rozmiaru danych. Zatem cała konstrukcja jednego kroku algorytmu redukcyjnego F jest wielomianowa.

Teraz zwróćmy uwagę na liczbę kroków którą potrzebuje zrealizować algorytm F. Przypomnę że cała idea konstrukcji polega na symulowaniu działania algorytmu A przez algorytm F. Zatem każda pojedyncza instrukcja algorytmu A przekłada się na jeden wielomianowy krok algorytmu F. Skoro więc wielomian od wielomianu nadal jest wielomianem czyli całość działania algorytmu F jest ostatecznie wielomianowa.

4. Podsumowanie

   Podsumowując pokazaliśmy, że algorytm redukcji F w poprawny sposób symuluje działanie każdego algorytmu dającego przedstawić się w strukturze komputerowej. Ponadto jeśli złożoność czasowa symulowanego algorytmu nie jest większa od wielomianowej to złożoność czasowa algorytmu redukcyjnego F też jest wielomianowa. Oznacza to, że problem CIRCUIT-SAT jest tak samo trudny jak każdy problem należący do klasy NP, czyli CIRCUIT-SAT $\in$ NP-trudny.

Twierdzenie 3. *CIRCUIT-SAT $\in$ NP-zupełny*

Dowód: Reasumując to że CIRCUIT-SAT $\in$ NP i CIRCUIT-SAT $\in$ NP-trudny otrzymujemy, że CIRCUIT-SAT $\in$ NP-zupełny.

Wynika z tego wniosek że o ile klasa NP-zupełna jest nie pusta (czyli ciągle nie rozstrzygnięte przez ludzkość pytanie czy NP = P) to niewątpliwie CIRCUIT-SAT należy do tej klasy.

Rozdział 3

Problem pokrycia wierzchołkowego

3.1. Wstęp

Tematem rozdziału będzie przedstawienie problemu minimalnego pokrycia wierzchołkowego. Nieformalnie pokryciem wierzchołkowym nazywamy zbioru wierzchołków, w którym każda krawędź z grafu ma co najmniej jeden koniec. Samo określenie problemu wydaje się być niezwykle proste, jednak rozwiązanie tego problemu jest NP-zupełne. Formalnie problem można zdefiniować następująco.

3.2. Przedstawienie problemu

Problem minimalnego pokrycia wierzchołkowego (VERTEX-COVER) polega na znalezieniu w grafie nieskierowanym $G = (V, E)$ takiego podzbioru $V' \subseteq V$, że dla każdej krawędzi $\{v, w\} \in E : v \in V' \vee w \in V'$.

Twierdzenie 4. *VERTEX-COVER ∈ NP-zupełnych.*

3.3. Dowód przez redukcje

Dowód:

W pierwszej kolejności należy pokazać, że VERTEX-COVER ∈ NP. Wejściem dla świadectwa jest graf $G = (V, E)$ oraz podzbiór wierzchołków $V' \subseteq V$. Zadnie polega na sprawdzeniu, czy każda krawędź ma co najmniej jeden koniec w zbiorze V'. Jeśli tak jest, to mamy do czynienia z poprawnym pokryciem wierzchołkowym. Nie trudno zauważyć, że czas potrzebny na to jest liniowy względem liczby krawędzi. Natomiast jeśli graf jest stosunkowo gęsty, to złożoność jest kwadratowa względem liczby wierzchołków. Tak czy inaczej jest wielomianowa w ujęciu rozmiaru danych, co za tym idzie VERTEX-COVER ∈ NP.

Teraz należy pokazać, że VERTEX-COVER ∈ NP-trudnych. W tym celu należy skorzystać z redukcji VERTEX-COVER do CLIQUE. Dopełniając graf posiadający klikę otrzymujemy graf, którego pokryciem wierzchołkowym jest wszystko poza kliką. Dopełnienie grafu należy rozumieć jako odwrócenie wszystkich krawędzi. Precyzując, wszystkie krawędzie które należały do grafu G nie należą do grafu dopełnionego $\bar{G}$, a te krawędzie które nie istniały w grafie G istnieje w dopełnionym grafie $\bar{G}$. W dalsze rozważania wykorzystują fakt, że suma krawędzi z grafu G i jego dopełnienia $\bar{G}$ daje graf pełny. Jak również to, że dopełnienie grafu dopełnionego daje graf pierwotny. Wszystko to razem oznacza, że problem maksymalnej kliki jest dualny z problemu minimalnego pokrycia wierzchołkowego. Zatem redukcja opierać się będzie na tezie: graf G zawiera klikę na wierzchołkach V' wtedy i tylko wtedy ,gdy $V - V'$ jest pokryciem wierzchołkowym dopełnionego grafu $\bar{G}$. W celu udo-

wodnienia tej tezy zostanie to rozbite na dwie implikacje.

W pierwszej kolejności zostanie pokazane, że jeśli graf G zawiera klikę na wierzchołkach V', to $V - V'$ jest pokryciem wierzchołkowym dopełnionego grafu $\bar{G}$. Dla wierzchołków $u, v \in V$ pomiędzy którymi nie istnieje krawędź w G zachodzi, że co najmniej jeden z nich nie należy do kliki, czyli należy do zbioru $V - V'$. Jako, że w grafie dopełnionym $\bar{G}$ pomiędzy wierzchołkami v i u istnieje krawędź to zbiór $V - V'$ pokryje ją. Jako, że krawędź została wybrana w sposób dowolny to rozumowanie jest poprawne dla każdej krawędzi z $\bar{G}$, co kończy dowodzenie pierwszej implikacji.

Teraz pozostało pokazać, że: jeśli $V - V'$ jest pokryciem wierzchołkowym dopełnionego grafu $\bar{G}$ to graf G zawiera klikę na V' wierzchołkach. Zwróćmy uwagę na to że zbiór V' zawiera wszystkie te wierzchołki które nie należą do pokrycia wierzchołkowego, zatem w grafie dopełnionym $\bar{G}$ między nimi nie występuje żadna krawędź. Zachodzi tak ponieważ jeśli którąś parę łączyła by krawędź to co najmniej jeden z tych wierzchołków musiał by należeć do pokrycia wierzchołkowego. Skoro w grafie dopełnionym $\bar{G}$ między zbiorem tych wierzchołków nie występuje krawędź to w grafie G między wszystkimi tymi wierzchołkami będą występowały krawędzie co z kolei oznacza że na zbiorze V' mamy klikę. To stwierdzenie kończy drugą implikację, ostatecznie wykazując prawdziwość tezy postawionej wcześniej.

Ostatnim krokiem jest określenie złożoności sprowadzenia VERTEX-COVER do CLIQUE. Przy takim postępowaniu należy wpierw stworzyć graf pełny na identycznej liczbie wierzchołków, by później iterując po każdej krawędzi pozostawiać jedynie te krawędzie które nie należały do pierwotnego grafu. Na koniec klika będzie znajdowała się na tych wierzchołkach które nie nie należały do pokrycia wierzchołkowego. Jak widać całość jest złożoności kwadratowej ze względu na ilość wierzchołków, czyli wielomianowa, co kończy

cały dowód wykazując że VERTEX-COVER $\in$ NP-zupełny.

3.4. Algorytm aproksymacyjny

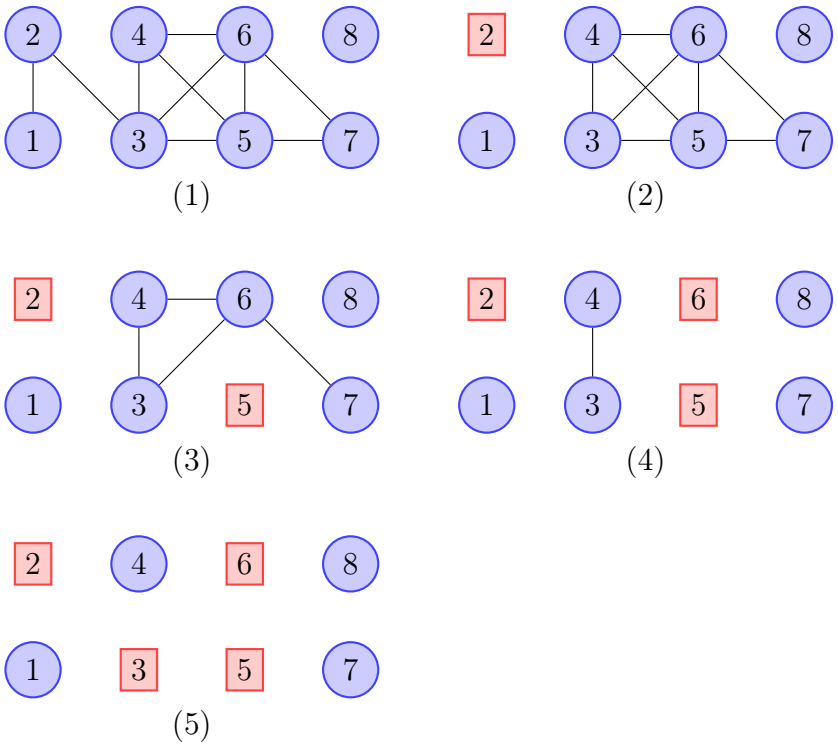
3.4.1. Algorytm VERTEX-COVER-DEGREES-APPROX

Można odnieść wrażenie że metoda zachłanna będzie przynosiła wyniki bardzo zbliżone do optymalnego, zatem spróbujmy w ten oto sposób podejść do problemu. Możemy wyróżnić dwa sposoby podejścia zachłannego. Pierwszy to taki, który pokrywał by wierzchołki o najwyższym stopniu, idąc za myślą że taki wierzchołek pokryje najwięcej krawędzi zatem naturalnie powinien wchodzić w skład pokrycia wierzchołkowego. Drugie podejście zachłanne to takie, w którym przyglądamy się końca(brzegom) grafu, czyli takim nie izolowanym wierzchołkom, które mają możliwie najmniejszy stopień wierzchołka. Tutaj naturalna jest myśl, że jeśli mamy pokryć te incydentne krawędzie to należało by do pokrycia wybrać wierzchołek z drugiego końca krawędzi. Obie metody w swojej idei wykorzystują najlepsze rozwiązanie lokalne co wydaje się prowadzić do dobrego przybliżenia problemu globalnego. W swoich rozważaniach skupie się na drugim podejściu, dlatego że to pierwsze jest popularnie rozważane w literaturze. Oto algorytm realizujący to podejście:

Algorithm 1 VERTEX-COVER-DEGREES-APPROX

```
1: function MyProcedure
2:     C ← ∅
3:     E' ← G.E
4:     while E' ≠ ∅ do
5:         weź dowolny nieizolowany wierzchołek v o najmniejszym stoppniu w grafie G
6:         weź incydentny wierzchołek u z wierzchołkiwm v o największym stopniu
7:         C ← C ∪ {u}
8:         usuń z E' incydentne krawędzie z u
9:     return C
```

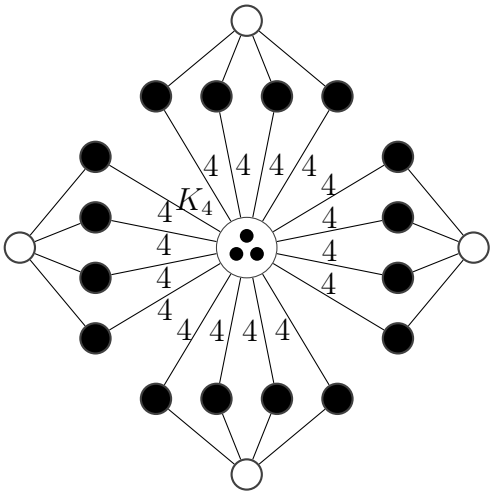
Rysunek 3.1 przedstawia kolejne kroki wykonywania algorytmu VERTEX-COVER-DEGREES-APPROX. W pierwszym kroku wierzchołek o najmniejszym stopniu to 1 a jedyny incydentny z nim wierzchołek który wchodzi do pokrycia to 2. W drugim kroku wierzchołek o najmniejszym stopniu to 7 z nim incydentne są 5 i 6. Oba są tego samego stopnia, zatem do pokrycia wejdzie dowolny z nich w tym przypadku 5. Dalej sytuacja jest klarowna i za pośrednictwem 7 do pokrycia wchodzi 6. Ostatni przebieg to pojedyncza krawędź w rezultacie do pokrycia wchodzi dowolnie wybrany 3. Reasumując pokrycie wierzchołkowe dla tego grafu wynosi 4 (wierzchołki: 2,3,5,6) i jest to optymalne rozwiązanie dla tego grafu.



Rys. 3.1. Przykład wykonania algorytmu VERTEX-COVER-DEGREES-APPROX

Zastanówmy się teraz nad konstrukcją grafu który będzie wpuklał tezę że pokrywanie wierzchołków wyższego stopnia daje gorsze rozwiązanie. Bardzo

dobrym przykładem jest graf, który nazwijmy płatkiem śniegu(snowflake). Taki graf składa się z rdzenia i wielu ramion. Poniżej zamieściłem schemat takiego grafu którego rdzeniem jest graf K_4 i składa się z 4 ramion. Możemy zauważyć że liczbę ramion dla takiego rdzenia może zbiegać do nieskończoności, natomiast liczba wierzchołków incydentnych do rdzenia nie może być więcej niż najniższy stopień wierzchołków do których przylegają. W miarę jak rdzeń grafu jest większy tym można do niego przyłączyć większe ramiona.



Rys. 3.2. Graf snowflake(4,4)

Wierzchołki oznaczone kolorem czarnym zostały nominowane do pokrycia wierzchołkowego grafu plus dodatkowo 3 wierzchołki z K_4 (nie można mniej dla grafu pełnego). Natomiast można łatwo zauważyć, że jeśli do pokrycia zaliczymy białe wierzchołki plus dodatkowo klikę, to również pokryje nam to graf przy znacznie mniejszej liczbie wierzchołków. Oszacujmy współczynnik aproksymacji dla tej rodziny grafów gdzie k będzie rozmiarem rdzenia a m liczbą ramion. Łatwo zauważyć że przypadek pesymistyczny to taki gdzie liczbę i rozmiar ramion będziemy maksymalizowali, natomiast rozmiar rdzenia będziemy minimalizowali. Aby własność doboru nieodpowiednich wierzchołków była zachowana to liczba wierzchołków połączonych z rdzeniem musi

być nie większa od rozmiaru rdzenia. Minimalny graf o jak największej liczbie krawędzi to graf pełny. W skład pokrycia optymalnego będzie wchodził zawsze cały rdzeń grafu, czyli k wierzchołków a dodatkowo m wierzchołków z każdego ramienia po tym wierzchołku który nie jest połączony z rdzeniem. W przypadku rozwiązania proponowanego przez algorytm mamy $k-1$ wierzchołków z rdzenia oraz $m * k$ wierzchołków ze wszsytkich ramion. W rezultacie otrzymujemy: $\frac{C}{C^*} = \frac{k-1+m*k}{k+m} \leq m * k$. Zatem zależność jest liniowa co świadczy o słabym współczynniku aproksymacji.

3.4.2. Algorytm VERTEX-COVER-EDGES-APPROX

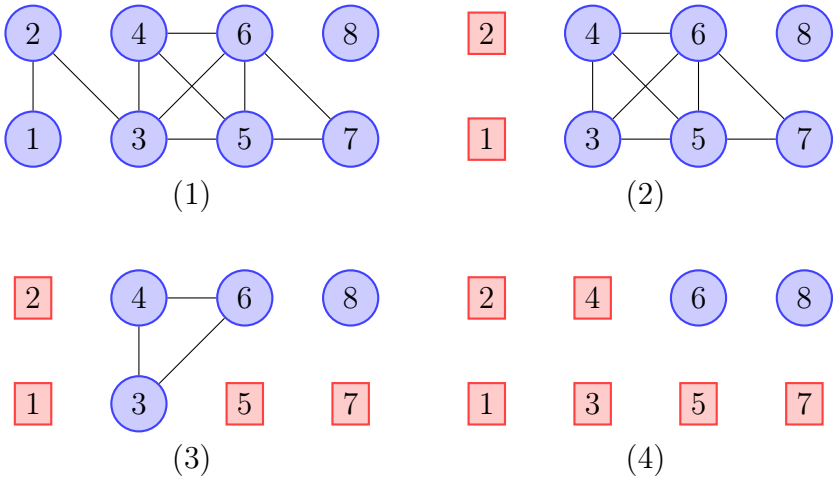
Przyjrzyjmy się teraz innemu podejściu do problemu. Proponowane rozwiązanie to wybrać losową krawędź, dwa jej końce dodać do pokrycia wierzchołkowego, po czym usunąć wszsytkie incydętne z nimi krawędzie. Oto algorytm realizujący to podejście:

Algorithm 2 VERTEX-COVER-EDGES-APPROX

```
1: function MyProcedure
2:     C ← ∅
3:     E' ← G.E
4:     while E' ≠ ∅ do
5:         weź dowolna krawędź(u, v) ∈ E'
6:         C ← C ∪ {u, v}
7:         usuń z E' incydentne krawędzie z u lub v
8:     return C
```

Teraz wykonajmy ten sam algorytm dla poprzedniego grafu. W pierwszym kroku w dowolny sposób wybraliśmy krawędź {1,2}, dodaliśmy te wierzchołki do pokrycia i usuneliśmy wszsytkie incydętne krawędzie. W drugim kroku analogicznie została wybrana dowolna krawędź {5,7}, wierzchołki dodane do pokrycia, a następnie usunięte wszystkie incydentne krawędzie. Całe rozumo-

wanie jest powtarzane aż w rezulatcie do pokrycia wierzchołkowego zostały przypisane wierzchołki: 1,2,3,4,5,7.



Rys. 3.3. Przykład wykonania algorytmu VERTEX-COVER-EDGES-APPROX

Porównując te dwa podejścia można odnieść wrażenie, że ostatni algorytm nie jest godny uwagi, lecz spróbójmy określić współczynnik aproksymacji dla algorytmu VERTEX-COVER-EDGES-APPROX.

Twierdzenie 5. *VERTEX-COVER-DEGREES-APPROX jest 2 aproksymowalne [do po polsku]*

Dowód: W pierwszej kolejności pokażmy że zbiór z wyjścia algorytmu VERTEX-COVER-EDGES-APPROX jest rzeczywiście pokryciem wierzchołkowym. Pętla z 4. linii wykonuje się do momentu aż wszystkie krawędzie nie zostaną pokryte. Wewnątrz w linii 5. do pokrycia dodajemy oba końce losowo wybranej krawędzi, następnie usuwając te krawędzie, które pokryte są przez dodane wierzchołki. W rezultacie zostają jedynie usunięte te krawędzie które są pokryte, a do zbioru wyjściowego wchodzą te wierzchołki przy których krawędź jeszcze nie została pokryta. W tym procesie otrzymany zbiór wierzchołków jest prawidłowym pokryciem wierzchołkowym dla zadanego grafu.

Przyjrzyjmy się teraz jakości tego rozwiązania. Algorytm VERTEX-COVER-EDGES-APPROX w linii 5. dodaje dwa wierzchołki z jednej krawędzi do pokrycia wierzchołkowego. W linii 7. zostają usunięte wszystkie incydentne krawędzie w rezultacie. Tak postępując można zaobserwować, że żadne losowo wybrane krawędzie nie mają wspólnego wierzchołka. Zatem zbiór zawiera tyle wierzchołków ile wynosi podwojona liczba losowych krawędzi. Zauważmy, że pokrycie optymalne musi analogicznie pokryć wszystkie krawędzie, w tym również te, które losowo wybraliśmy. W rezultacie pokrycie generowane przez algorytm VERTEX-COVER-EDGES-APPROX jest nie więcej niż 2 krotnie większe od optymalnego.

Choć algorytm VERTEX-COVER-EDGES-APPROX posiada znacząco niższy współczynnik aproksymacji od algorytmu VERTEX-COVER-DEGREES-APPROX to widać, że swoją przewagę zawdzięcza jedynie szczególnym przypadkom. W ogólności algorytm VERTEX-COVER-EDGES-APPROX będzie zwracał gorsze rozwiązanie od algorytmu VERTEX-COVER-DEGREES-APPROX.

Rozdział 4

Problem kliki

4.1. Wstęp

Kliką nazywamy taki podzbiór wierzchołków V' ze zbioru V, że występujące w nim wszystkie wierzchołki są połączone między sobą krawędziami. Jeżeli moc zbioru $|V'| = k$ to mówimy że klika jest rozmiaru k, oczywiście każdy graf pełny K_n jest kliką rozmiaru n.

4.2. Przedstawienie problemu

Problem maksymalnej kliki (CLIQUE) polega na znalezieniu w grafie nieskierowanym G kliki o największym rozmiarze.

Twierdzenie 6. *$CLIQUE \in$ NP-zupełnych.*

4.3. Dowód przez redukcje

Dowód: Sprawdzenie, że problemu CLIQUE należy do klasy NP odbywa się poprzez sprawdzenie podzbioru V'[zawarty bądź równy] V będącego świadectwem. Jeśli w podzbiorze V' między każdą parą wierzchołków występuje krawędź to oznacza to że tworzą one klikę. Taki algorytm sprawdzający wykonuje dokładnie $teta - ozkreska(n) = n^2$ operacji dla każdego wierzchołka, czyli jest złożoności wielomianowej. Dowodzi to że CLIQUE [należy do] NP.

Kolejnym krokiem dowodu jest pokazanie że problem CLIQUE jest Np-trudny. Aby tego dokonać wykorzystam fakt że problem 3-CNF jest NP-zupełny. Problem 3-CNF jest formułą logiczną zawierającą klauze połączone koniunkcją, natomiast każda klauzula składa się z dokładnie trzech literałów połączonych alternatywą. Zadanie polega na znalezieniu takich wartościowań literałów dzięki których formuła jest spełniana.

Przedstawię teraz sposób sprowadzenia problemu 3-CNP do problemu znalezienia kliki w grafie. Każda klauzula będzie określała oddzielną grupę wierzchołków, a w każdej z nich będą się znajdowały dokładnie trzy wierzchołki odpowiadające konkretnym literałom. Krawędzią połączymy te wierzchołki które spełniają jednocześnie dwa warunki: - wierzchołki nie należą do wspólnej klauzuli - literały odpowiadające wierzchołkom nie są sprzeczne. Szukać będziemy takiej kliki która połączy ze sobą wszystkie klauzule. Jak łatwo zauważyć transformacja egzemplarza 3-CNF na egzemplarz grafowy można bez większych problemów wykonać w czasie wielomianowym.

Przyjrzyjmy się teraz związkowi jaki tworzą między sobą te dwa zadania. Nie połączyliśmy ze sobą wierzchołków w tej samej klauzuli co oznacza że wystarczy że znajdziemy tylko jeden literał prawdziwy, co sprawi że dzięki wewnętrznym alternatywą cała klauzula będzie prawdziwa. Łączymy ze so-

bą tylko nie sprzeczne literały co oznacza że zmienne nie mogą jednocześnie posiadać dwóch różnych wartości. Zbierając to wszystko do siebie chcę pokazać że znalezienie kliki na tych klauzulach oznacza znalezienie niesprzecznych wartości literałów które w rezultacie spełniają całą formułę. Natomiast z drugiej strony znalezienie wartościowań literałów pozwalających uczynić formułę spełnialną oznacza że w takiej konstrukcji grafu musi istnieć klika wielkości liczbie klauzul w formule.

——————————— Świadectwem dla problemu CLIQUE jest podzbiór wierzchołków V' zawierających klikę. Aby sprawdzić autentyczność tego świadectwa wystarczy jedynie sprawdzić, czy dla każdej pary wierzchołków z tego podzbioru istnieje krawędź. To sprawdzenie ma złożoność kwadratową, czyli wielomianową.

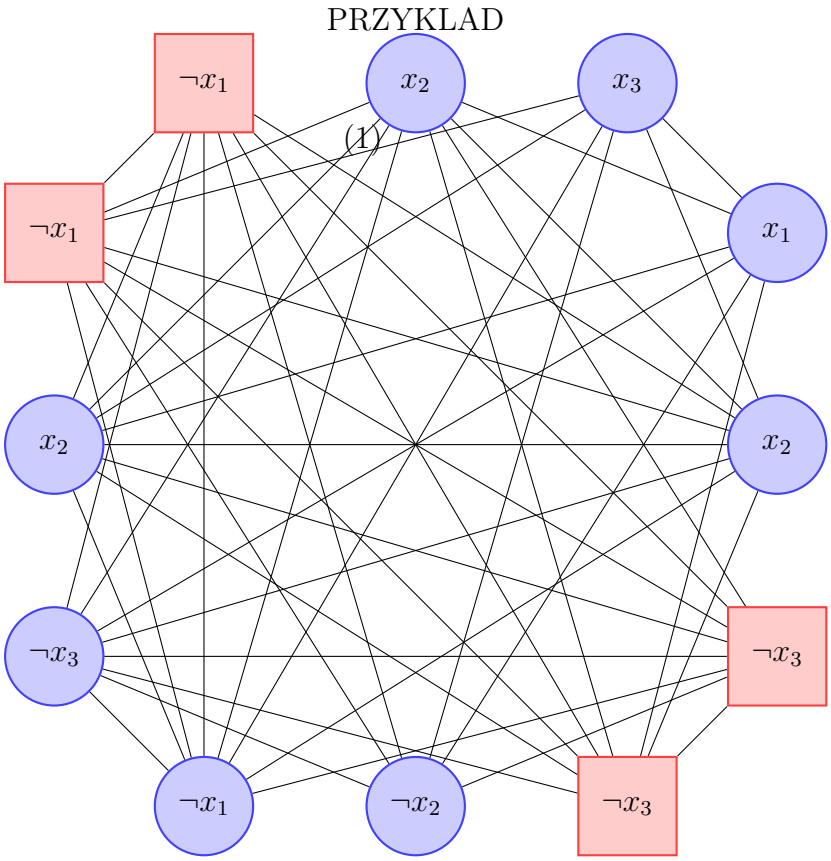
Aby pokazać, że CLIQUE jest NP-trudny podeprzemy się faktem że problem 3-CNF jest NP-trudny i przedstawię redukcje 3-CNF [<=] p CLIQUE. [kolki = C1 i C2 i ... Ck] oznaczmy jak formułę problemu 3-CNF składającą się z $r = 1, 2, ...k$ klauzul. Na każdą klauzulę składają się dokładnie 3 literały $l^r, 1, l^r, 2, l^r, 3$. Skonstruujemy graf G z formuły [kolko] w ten sposób że formuła jest spełniana wtedy i tylko wtedy, gdy G zawiera klikę o rozmiarze k.

... redukcja z 3CNF

koniec

4.4. Algorytm aproksymacyjny

Dzięki temu że problem CLIQUE jest dualny do problemu VERTEX-COVER, to algorytmy z poprzedniego rozdziału również po odpowiednim przeformułowaniu wejścia zwracają rozwiązania problemu CLIQUE



Rys. 4.1. Przykład wykonania algorytmu VERTEX-COVER-EDGES-APPROX

Rozdział 5

Problem komiwojażera

co to jest ?

5.1. Wstęp

5.2. Przedstawienie problemu

5.3. Dowód przez redukcje

5.4. Algorytm aproksymacyjny

Rozdział 6

Problem sumy podzbioru

6.1. Wstęp

6.2. Przedstawienie problemu

6.3. Dowód przez redukcje

6.4. Algorytm aproksymacyjny

Spis rysunków

Spis tabel