1 Podstawowe pojęcia

1.1 Co to jest programowanie liniowe?

Program liniowy (w skrócie PL[1]) jest to problem minimalizacji/maksymalizacji liniowej *funkcji celu* o n argumentach $x_1, x_2, \ldots, x_n$ przy zachowaniu pewnej liczby równości lub nierówności liniowych (będziemy je nazywać *ograniczeniami*) zawierających zmienne x_i.

Przykład:

$$\begin{array}{ll} \text{zminimalizuj} & x_1 + 2x_2 \\ \text{z zachowaniem warunków} & x_2 \leq x_1 + 2 \\ & 2x_1 + x_2 \leq 4 \\ & 2x_2 + x_1 \geq 0 \\ & x_2 \geq 0 \end{array}$$

Za pomocą programów liniowych można wyrazić bardzo wiele naturalnych problemów optymalizacyjnych. Dla przykładu, problem maksymalnego przepływu w sieci o n wierzchołkach, źródle s, ujściu t, i funkcji przepustowości $c : V^2 \to \mathbb{R}$ można wyrazić za pomocą następującego programu liniowego o $|V|^2$ zmiennych (dla każdej pary wierzchołków v, w mamy zmienne f_{vw} i f_{wv} odpowiadające przepływowi od v do w i odwrotnie) i $2|V|^2 + |V| - 2$ ograniczeniach:

$$\begin{array}{lll} \text{zmaksymalizuj} & \sum_{v \in V} f_{sv} & \\ \text{z zachowaniem warunków} & f_{vw} \leq c(v,w) & v, w \in V \\ & f_{vw} = -f_{wv} & v, w \in V \\ & \sum_{w \in V} f_{vw} = 0 & v \in V \setminus \{s, t\} \end{array}$$

Inny przykład: znajdowanie długości najkrótszej ścieżki od s do t w grafie $G = (V, E)$ z funkcją $w : E \to \mathbb{R}$ opisującą długości krawędzi. Tu dla każdego wierzchołka v mamy zmienną d_v. Dla wierzchołków na najkrótszej ścieżce od s do t zmienna d_v będzie odpowiadać odległości od s do v.

$$\begin{array}{lll} \text{zmaksymalizuj} & d_t & \\ \text{z zachowaniem warunków} & d_v \leq d_u + w(u,v) & (u,v) \in E \\ & d_s = 0 & \end{array}$$

Zadanie. Udowodnij, że ten program faktycznie jest równoważny problemowi najkrótszej ścieżki.
Rozwiązanie. Dowód można przedstawić w następujący nieformalny sposób: wyobraźmy sobie że mamy fizyczny model grafu, w którym wierzchołki (powiedzmy metalowe kulki) są połączone sznurkami o odpowiednich długościach. Żeby znaleźć odległość od s do t chwytamy za kulki odpowiadające tym wierzchołkom i staramy się je maksymalnie od siebie oddalić (aż do całkowitego napięcia niektórych sznurków).

[1] w tekstach anglojęzycznych powszechnie używany jest skrót LP

1.2 Terminologia, postać kanoniczna i dopełnieniowa

Rozważmy PL o n zmiennych $x_1, \ldots, x_n$. Wartości zmiennych możemy utożsamiać z wektorem $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$. Zauważmy też, że liniową funkcję celu $\sum_{j=1}^{n} c_j x_j$ możemy zapisać krócej jako $\mathbf{c}^T\mathbf{x}$ dla wektora $\mathbf{c} = (c_1, \ldots, c_n)$. Będziemy utożsamiać tę funkcję z wektorem $\mathbf{c}$.

- $\mathbf{x} \in \mathbb{R}^n$ jest *rozwiązaniem dopuszczalnym*[2] PL gdy $\mathbf{x}$ spełnia wszystkie ograniczenia.
- $\mathbf{x} \in \mathbb{R}^n$ jest *rozwiązaniem optymalnym* PL gdy $\mathbf{x}$ jest rozwiązaniem dopuszczalnym i optymalizuje funkcję celu, tzn. jeśli dla dowolnego dopuszczalnego $\mathbf{y} \in \mathbb{R}^n$ jest $\mathbf{c}^T\mathbf{x} \leq \mathbf{c}^T\mathbf{y}$ w przypadku gdy PL jest minimalizacyjny ($\mathbf{c}^T\mathbf{x} \geq \mathbf{c}^T\mathbf{y}$ gdy maksymalizacyjny).
- PL jest *dopuszczalny* gdy istnieje rozwiązanie dopuszczalne, w przeciwnym przypadku jest *sprzeczny*[3].
- PL jest *nieograniczony* gdy jest dopuszczalny ale nie ma rozwiązań optymalnych, tzn. dla PL minimalizacyjnego dla dowolnego $\lambda \in \mathbb{R}$ istnieje rozwiązanie dopuszczalne $\mathbf{x}$ takie, że $\mathbf{c}^T\mathbf{x} < \lambda$.

Będziemy posługiwać się trzema wygodnymi postaciami programów liniowych: kanoniczną, standardową i dopełnieniową[4].

1.2.1 Postać kanoniczna

Postać kanoniczna jest najbardziej ogólna z trzech rozważanych tu postaci. Program w postaci kanonicznej wygląda następująco:

$$\begin{array}{ll} \text{zmaksymalizuj} & \sum_{j=1}^{n} c_j x_j \\ \text{z zachowaniem warunków} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \qquad i = 1, \ldots, m \end{array}$$

gdzie $\{a_{ij}, b_i, c_j\}$ są dane. Wygodniej będzie nam używać zapisu macierzowego:

$$\begin{array}{ll} \text{zmaksymalizuj} & \mathbf{c}^T\mathbf{x} \\ \text{z zachowaniem warunków} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{array}$$

gdzie $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{A} \in M_{m \times n}[\mathbb{R}]$.

Lemat 1. *Każdy PL można sprowadzić do postaci kanonicznej.*

Dowód. Zadanie min $\mathbf{c}^T\mathbf{x}$ zamieniamy na max $(-\mathbf{c})^T\mathbf{x}$. Równości $\mathbf{a}_i^T\mathbf{x} = b_i$ zamieniamy na parę $\mathbf{a}_i^T\mathbf{x} \leq b_i$, $\mathbf{a}_i^T\mathbf{x} \geq b_i$. Nierówności $\mathbf{a}_i^T\mathbf{x} \geq b_i$ zamieniamy na $(-\mathbf{a}_i)^T\mathbf{x} \leq -b_i$. □

[2] ang. feasible solution

[3] ang. infeasible

[4] W literaturze panuje niemały bałagan dotyczący terminologii postaci PL. W szczególności, w wielu tekstach postać dopełnieniowa (ang. augmented) jest nazywana standardową (standard). Dokonaliśmy tu takiego wyboru, ponieważ słowo „dopełnieniowa" więcej mówi o naturze tej postaci, natomiast większość naturalnych problemów algorytmicznych zapisuje się standardowo w postaci, którą nazwiemy standardową. Wreszcie, ta terminologia jest używana w Cormenie i Vaziranim

1.2.2 Postać standardowa

Programy liniowe modelujące problemy algorytmiczne bardzo często są w następującej postaci (jest to szczególny przypadek postaci kanonicznej):

$$\begin{array}{ll} \text{zmaksymalizuj} & \mathbf{c}^T\mathbf{x} \\ \text{z zachowaniem warunków} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

Lemat 2. *Każdy PL można sprowadzić do postaci standardowej.*

Dowód. Możemy założyć, że mamy już PL w postaci kanonicznej. Aby zapewnić nieujemność każdej zmiennej, dla każdej zmiennej x_i wprowadzamy parę zmiennych x_i^+ i x_i^- i każde wystąpienie x_i w PL (także w funkcji celu) zastępujemy przez $x_i^+ - x_i^-$. Dodajemy też $x_i^+ \geq 0$ i $x_i^- \geq 0$. □

Czasem nie chcemy sztucznie zamieniać problemu minimalizacji na maksymalizację przez odwrócenie funkcji celu. Postać standardowa w wersji minimalizacyjnej wygląda następująco:

$$\begin{array}{ll} \text{zminimalizuj} & \mathbf{c}^T\mathbf{x} \\ \text{z zachowaniem warunków} & \mathbf{A}\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array}$$

1.2.3 Postać dopełnieniowa

Program w postaci dopełnieniowej (w wersji maksymalizacyjnej) wygląda następująco:

$$\begin{array}{lll} \text{zmaksymalizuj} & \mathbf{c}^T\mathbf{x} & \\ \text{z zachowaniem warunków} & \mathbf{A}\mathbf{x} = \mathbf{b} & i = 1, \ldots, m \\ & \mathbf{x} \geq \mathbf{0} & \end{array}$$

gdzie $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m, \mathbf{A} \in M_{m\times n}[\mathbb{R}]$. Zauważmy, że z dokładnością do zamiany równości na pary nierówności możemy powiedzieć, że PL w postaci dopełnieniowej jest w postaci standardowej.

Lemat 3. *Każdy PL można sprowadzić do postaci dopełnieniowej.*

Dowód. Możemy założyć, że mamy już PL w postaci standardowej. Dla każdej nierówności $\mathbf{a}_i^T\mathbf{x} \geq b_i$ wprowadzamy „zmienną dopełnieniową”[5] s_i i nierówność zastępujemy przez $\mathbf{a}_i^T\mathbf{x} - s_i = b_i$ oraz $s_i \geq 0$. □

[5] ang. slack variable

2 Geometria programów liniowych

Przypomnijmy:

- zbiór punktów spełniających $\mathbf{a}^T\mathbf{x} = b$, $\mathbf{a} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}$ to *hiperpłaszczyzna*,
- zbiór punktów spełniających $\mathbf{a}^T\mathbf{x} \geq b$ to *półprzestrzeń*,
- zbiór rozwiązań dopuszczalnych PL w postaci kanonicznej $\mathbf{Ax} \leq b$ to przecięcie półprzestrzeni, czyli *wielościan*[6].

Dla przykładu, zbiór rozwiązań dopuszczalnych PL z rozdziału 1.1 jest czworokątem (w 2 wymiarach hiperpłaszczyzny to proste, półprzestrzenie to półpłaszczyzny, a wielościany to wielokąty). Zauważmy, że rozwiązanie optymalne to najdalszy punkt wielościanu w kierunku wektora funkcji celu (dla problemu maksymalizacyjnego; dla minimalizacyjnego w kierunku wektora odwrotnego).

Zauważmy, że zbiór rozwiązań PL jest przecięciem zbiorów wypukłych (półprzestrzeni, hiperpłaszczyzn) a więc jest wypukły.

2.1 Struktura rozwiązań optymalnych

Zdefiniujemy teraz trzy naturalne pojęcia związane z programami liniowymi i ich geometryczną interpretacją. Za chwilę pokażemy, że są one równoważne.

- *Wierzchołkiem* wielościanu $\mathcal{P}$ nazywamy dowolny punkt $\mathbf{x} \in \mathcal{P}$, który jest jedynym rozwiązaniem optymalnym dla pewnej funkcji celu $\mathbf{c}$.
- *Punktem ekstremalnym* wielościanu $\mathcal{P}$ nazywamy dowolny punkt $\mathbf{x} \in \mathcal{P}$, który nie jest wypukłą kombinacją dwóch innych punktów $\mathbf{y}, \mathbf{z} \in \mathcal{P}$. (Przypomnijmy, że wypukła kombinacja $\mathbf{y}$ i $\mathbf{z}$ to dowolny punkt postaci $\lambda\mathbf{y} + (1-\lambda)\mathbf{z}$ dla pewnego $\lambda \in [0,1]$.)
- *Bazowe rozwiązanie dopuszczalne* (brd) PL o n zmiennych to rozwiązanie dopuszczalne $\mathbf{x} \in \mathbb{R}^n$ takie, że istnieje n liniowo niezależnych ograniczeń (w sensie wektorów współczynników przy x_i, tzn. $x_1+2x_2 \geq 1$ i $x_1+2x_2 \leq 2$ są liniowo zależne), które dla $\mathbf{x}$ są spełnione z równością[7]. Na przykład $(0,0)$ i $(\frac{2}{3}, \frac{8}{3})$ są brd programu z rozdziału 1.1.

W poniższych lematach rozważamy dowolny program liniowy oznaczamy przez $\mathcal{P}$ wielościan jego rozwiązań dopuszczalnych. Zakładamy, że jest on dany w postaci $\max \mathbf{c}^T\mathbf{x}$, $\mathbf{Ax} \leq \mathbf{b}$ (sprowadzenie dowolnego programu do takiej postaci nie zmienia wielościanu rozwiązań dopuszczalnych).

Lemat 4. *Jeśli* $\mathbf{x}$ *jest wierzchołkiem* $\mathcal{P}$ *to* $\mathbf{x}$ *jest punktem ekstremalnym.*

[6] ang. polyhedron
[7] ang. tight

Dowód. Niech $\mathbf{c}$ będzie funkcją celu taką, że $\mathbf{x}$ jest jedynym rozwiązaniem optymalnym LP dla funkcji celu $\mathbf{c}$.

Załóżmy, że $\mathbf{x} = \lambda\mathbf{y} + (1-\lambda)\mathbf{z}$ dla pewnych $\mathbf{y}, \mathbf{z} \in \mathcal{P}$ oraz $\lambda \in [0,1]$. Ponieważ $\mathbf{x}$ jest jedyny optymalny więc $\mathbf{c}^T\mathbf{y}, \mathbf{c}^T\mathbf{z} < \mathbf{c}^T\mathbf{x}$. Ale wówczas, z liniowości $\mathbf{c}$

$$\mathbf{c}^T\mathbf{x} = \lambda\mathbf{c}^T\mathbf{y} + (1-\lambda)\mathbf{c}^T\mathbf{z} < \lambda\mathbf{c}^T\mathbf{x} + (1-\lambda)\mathbf{c}^T\mathbf{x} = \mathbf{c}^T\mathbf{x}.$$

□

Lemat 5. *Jeśli* $\mathbf{x}$ *jest punktem ekstremalnym* $\mathcal{P}$ *to* $\mathbf{x}$ *jest bazowym rozwiązaniem dopuszczalnym.*

Dowód. Z definicji punktu ekstremalnego $\mathbf{x}$ jest dopuszczalny. Załóżmy, że nie jest brd, tzn. że nie istnieje n liniowo niezależnych ograniczeń spełnionych z równością dla $\mathbf{x}$. Intuicyjnie, oznacza to, że wokół $\mathbf{x}$ jest nieco „luzu", tzn. jest pewien wektor $\mathbf{d}$ (liniowo niezależny od wektorów ograniczeń spełnionych z równością), wzdłuż którego możemy się przemieszczać z $\mathbf{x}$ w przód i w tył pozostając w $\mathcal{P}$. Istotnie, pokażemy, że $\mathbf{x}$ jest kombinacją wypukłą dwóch punktów w $\mathcal{P}$.

Niech $T = \{i \mid \mathbf{a}_i^T\mathbf{x} = b_i\}$. Wiemy, że $\{\mathbf{a}_i \mid i \in T\}$ nie rozpina $\mathbb{R}^n$, tzn. $\operatorname{rank}[a_i, i \in T] < n$. Stąd układ równań (jednorodny)

$$\mathbf{a}_i^T\mathbf{d} = 0 \quad i \in T$$

ma przestrzeń rozwiązań wymiaru $n - \operatorname{rank}[\mathbf{a}_i \mid i \in T] \geq 1$. Czyli istnieje $\mathbf{d} \neq \mathbf{0}$ który jest rozwiązaniem tego układu. Pokażemy, że dla pewnego małego $\epsilon > 0$, $x \pm \epsilon\mathbf{d} \in \mathcal{P}$, czyli będzie sprzeczność (bo $\mathbf{x}$ jest ekstremalny). Istotnie, jeśli $i \in T$ to dla dowolnego ϵ, $\mathbf{a}_i^T(\mathbf{x} \pm \epsilon\mathbf{d}) = b_i$. Dla $i \notin T$ mamy $\mathbf{a}_i^T\mathbf{x} > b_i$, a więc na pewno można wybrać dostatecznie małe ϵ, żeby $\mathbf{a}_i^T(\mathbf{x} \pm \epsilon\mathbf{d}) \geq b_i$ dla każdego $i \notin T$. □

Lemat 6. *Jeśli* $\mathbf{x}$ *jest bazowym rozwiązaniem dopuszczalnym PL to* $\mathbf{x}$ *jest wierzchołkiem* $\mathcal{P}$.

Dowód. Niech $T = \{i \mid \mathbf{a}_i^T\mathbf{x} = b_i\}$. Podamy funkcję celu $\mathbf{c}$, przy której $\mathbf{x}$ jest jedynym rozwiązaniem optymalnym: $\mathbf{c} = \sum_{i\in T} \mathbf{a}_i$. Dla dowolnego $\mathbf{y} \in \mathcal{P}$ mamy

$$\mathbf{c}^T\mathbf{y} = \sum_{i\in T} \mathbf{a}_i^T\mathbf{y} \leq \sum_{i\in T} b_i = \mathbf{c}^T\mathbf{x},$$

(czyli $\mathbf{x}$ jest rozwiązaniem optymalnym) przy czym równość zachodzi tylko gdy dla każdego $i \in T$, $\mathbf{a}_i^T\mathbf{y} = b_i$, a to jest układ którego jedynym rozwiązaniem jest x (bo $\operatorname{rank}[\mathbf{a}_i \mid i \in T] = n$). □

Wniosek 1. *Dla dowolnego programu liniowego są równoważne:*

(i) $\mathbf{x}$ *jest wierzchołkiem,*

(ii) $\mathbf{x}$ *jest punktem ekstremalnym,*

(iii) $\mathbf{x}$ *jest bazowym rozwiązaniem dopuszczalnym.* □

Twierdzenie 1. *Każdy ograniczony PL w postaci standardowej* $\max \mathbf{c}^T\mathbf{x}$, $\mathbf{A}\mathbf{x} \leq \mathbf{b}$, $\mathbf{x} \geq \mathbf{0}$ *ma rozwiązanie optymalne, które jest punktem ekstremalnym.*

Dowód. Niech $\mathbf{x}$ będzie rozwiązaniem optymalnym PL. Jeśli $\mathbf{x}$ jest ekstremalny – koniec. W przeciwnym przypadku pokażemy jak przejść od $\mathbf{x}$ do punktu ekstremalnego o tej samej wartości funkcji celu (używając analogii trójwymiarowej, przesuniemy się z $\mathbf{x}$ wewnątrz wielościanu do ściany wielościanu, następnie ze środka ściany do krawędzi, a z krawędzi do wierzchołka).

Oznaczmy przez $\mathcal{P}$ wielościan rozwiązań dopuszczalnych. Skoro $\mathbf{x}$ nie jest punktem ekstremalnym, to istnieje $\mathbf{y} \neq \mathbf{0}$ taki że $\mathbf{x}+\mathbf{y}, \mathbf{x}-\mathbf{y} \in \mathcal{P}$.

Po pierwsze zauważmy, że przesuwając się wzdłuż wektora $\mathbf{y}$ nie zmieniamy wartości funkcji celu. Istotnie, gdyby $\mathbf{c}^T\mathbf{y} > \mathbf{0}$ to $\mathbf{c}^T(\mathbf{x}+\mathbf{y}) > \mathbf{c}^T\mathbf{x}$, natomiast gdyby $\mathbf{c}^T\mathbf{y} < \mathbf{0}$ to $\mathbf{c}^T(\mathbf{x}-\mathbf{y}) > \mathbf{c}^T\mathbf{x}$, czyli w obu przypadkach dostajemy sprzeczność z założniem że $\mathbf{x}$ jest optymalny. Skoro $\mathbf{c}^T\mathbf{y} = \mathbf{0}$ to dla dowolnego α, $\mathbf{c}^T(\mathbf{x}+\alpha\mathbf{y}) = \mathbf{c}^T\mathbf{x}$.

Rozważmy teraz dowolne ograniczenie spełnione z równością, czyli $(\mathbf{a}_i)^T\mathbf{x} = \mathbf{b}_i$. Teraz zauważmy, że skoro $(\mathbf{a}_i)^T(\mathbf{x}+\mathbf{y}) \leq \mathbf{b}$ to $(\mathbf{a}_i)^T\mathbf{y} \leq 0$. Podobnie skoro $(\mathbf{a}_i)^T(\mathbf{x}-\mathbf{y}) \leq \mathbf{b}$ to $(\mathbf{a}_i)^T\mathbf{y} \geq 0$. Stąd $(\mathbf{a}_i)^T\mathbf{y} = \mathbf{0}$ a nawet $(\mathbf{a}_i)^T(\alpha\mathbf{y}) = \mathbf{0}$ dla dowolnego $\alpha \in \mathbb{R}$. Czyli jeśli i-te ograniczenie jest spełnione z równością, to po przesunięciu się wzdłuż $\mathbf{y}$ dalej tak będzie, tzn. $(\mathbf{a}_i)^T(\mathbf{x}+\alpha\mathbf{y}) = \mathbf{b}_i$. Geometrycznie, odpowiada to spostrzeżeniu, że jeśli $\mathbf{x}$ jest na krawędzi (ścianie, ...), to posuwając się wzdłuż $\mathbf{y}$ dalej pozostajemy na krawędzi (ścianie, ...).

Niech teraz $\lambda = \max\{\alpha \mid \mathbf{x}+\alpha\mathbf{y} \in \mathcal{P} \text{ oraz } \mathbf{x}-\alpha\mathbf{y} \in \mathcal{P}\}$. Weźmy j takie, że $y_j \neq 0$. Zauważmy, że jeśli $x_j = 0$ to $y_j = 0$ (wpp. $x_j + y_j < 0$ lub $x_j - y_j < 0$, czyli $\mathbf{x}+\mathbf{y} \notin \mathcal{P}$ lub $\mathbf{x}-\mathbf{y} \notin \mathcal{P}$), a więc musi być $x_j \neq 0$. Oznacza to, że dla dostatecznie dużego α, $x_j + \alpha y_j = 0$ lub $x_j - \alpha y_j = 0$, czyli λ jest dobrze określone. Zauważmy, że w rozwiązaniu dopuszczalnym $\mathbf{x}+\lambda\mathbf{y}$ *rośnie* liczba ograniczeń spełnionych z równością. To implikuje, że powyższą operację wykonamy co najwyżej $n+m$ razy zanim dojdziemy do punktu ekstremalnego. □

Wniosek 2. *Istnieje algorytm (brutalny), który rozwiązuje PL w postaci standardowej o n zmiennych i m ograniczeniach w czasie $O(\binom{m}{n}n^3)$.*

Dowód. Żeby rozwiązać LP w postaci standardowej wystarczy sprawdzić wszystkie wierzchołki wielościanu i wybrać wierzchołek o najmniejszej wartości funkcji celu. Wierzchołków jest tyle co bazowych rozwiązań dopuszczalnych, czyli $\binom{m}{n}$. Aby znaleźć taki wierzchołek wystarczy rozwiązać układ odpowiednich n liniowo niezależnych równań. □

(Uwaga. W powyższym twierdzeniu m oznacza liczbę ograniczeń, a *nie* liczbę wierszy macierzy $\mathbf{A}$. Jeśli tę liczbę wierszy oznaczymy przez r to $m = r+n$).

3 Algorytmy programowania liniowego

Pierwszym algorytmem programowania liniowego był algorytm simplex, opublikowany przez George'a Dantziga w 1947 roku. Algorytm ten najpierw znajduje pewien wierzchołek wielościanu rozwiązań dopuszczalnych, a następnie w pętli przemieszcza się wzdłuż krawędzi do jednego z sąsiednich wierzchołków tak, aby poprawić wartość funkcji celu. Niestety okazuje się, że jego pesymistyczna złożoność jest wykładnicza. Doskonale zachowuje się on jednak dla „rzeczywistych" danych i jest powszechnie stosowany w praktyce.

Kolejny przełom nastąpił w 1979 kiedy to Leonid Khachiyan opublikował tzw. metodę elipsoidalną, czyli algorytm wielomianowy o złożoności $O(n^4L)$, gdzie L jest ograniczone z góry przez długość zapisu binarnego danych (macierzy $\mathbf{A}$, wektorów $\mathbf{b}$ i $\mathbf{c}$). Istnieją implementacje tego algorytmu, jednakże źle sprawdzają się w praktyce.

W 1984 roku zupełnie inne podejście zaproponował Narendra Karmarkar. Jego metoda punktu wewnętrznego osiąga złożoność $O(n^{3.5}L)$ i została poprawiona przez kolejnych autorów do $O(n^3L)$. Podobno niektóre implementacje niektórych wersji tego algorytmu zachowują się bardzo dobrze dla rzeczywistych danych (porównywalnie albo lepiej niż algorytm simplex). Mimo to, nawet ten algorytm jest rzadko stosowany praktyce.

3.1 Algorytm simplex

W poprzednim rozdziale zauważyliśmy już, że poszukując rozwiązań optymalnych można ograniczyć się do wierzchołków wielościanu. Algorytm simplex korzysta z tej obserwacji i realizuje podejście *local search*. Dokładniej, algorytm zaczyna od dowolnego wierzchołka wielościanu i w każdej kolejnej iteracji próbuje przemieścić się do takiego sąsiedniego wierzchołka, że wartość funkcji celu poprawia się (lub przynajmniej nie pogarsza).

Opiszemy teraz algorytm simplex na przykładzie konkretnego programu liniowego:

$$\begin{aligned} \max \quad & 3x_1 + x_2 + 2x_3 \\ & x_1 + x_2 + 3x_3 \leq 30 \\ & 2x_1 + 2x_2 + 5x_3 \leq 24 \\ & 4x_1 + x_2 + 2x_3 \leq 36 \\ & x_1, x_2, x_3 \geq 0. \end{aligned}$$

Zauważmy, że jest to program w postaci standardowej (w wersji maksymalizacyjnej), oraz wszystkie wyrazy wolne z prawych stron nierówności są nieujemne. Dzięki temu łatwo znaleźć dla niego rozwiązanie dopuszczalne — rozwiązanie zerowe: $x_1 = 0$, $x_2 = 0$, $x_3 = 0$.

Zapiszmy teraz ten program w postaci dopełnieniowej, wprowadzając zmienną dopełnieniową dla każdej nierówności (z wyłączeniem warunków nieujemnościowych). Dodatkowo funkcję celu zastąpmy nową zmienną z:

$$\begin{aligned} \max \quad & z \\ z &= 3x_1 + x_2 + 2x_3 \\ x_4 &= 30 - x_1 - x_2 - 3x_3 \\ x_5 &= 24 - 2x_1 - 2x_2 - 5x_3 \\ x_6 &= 36 - 4x_1 - x_2 - 2x_3 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0. \end{aligned}$$

Nasze rozwiązanie dopuszczalne, rozszerzone o zmienne dopełnieniowe ma teraz następującą postać:

$$x_1 = 0, x_2 = 0, x_3 = 0, \quad x_4 = 30, x_5 = 24, x_6 = 36.$$

Podczas działania algorytmu, w kolejnych krokach będziemy zmieniać nasz *program liniowy*. Mimo, iż ograniczenia będą się zmieniać, zawsze będą one opisywać ten sam wielościan (tzn. zbiór rozwiązań dopuszczalnych). W każdym kroku nasz program liniowy będzie miał szczególną postać, która w sposób jednoznaczny będzie wyznaczać pewien wierzchołek wielościanu — najlepsze dotąd znalezione rozwiązanie. Podamy teraz niezmiennik, który opisuje postać tych programów.

Niezmiennik sformułujemy dla dowolnego programu (a nie tylko powyższego przykładu). Załóżmy, że początkowy program w postaci dopełnieniowej ma m równości, oraz zawiera zmienne $x_1, \ldots, x_{n+m}$ (gdzie $x_{n+1}, \ldots, x_{n+m}$ są zmiennymi dopełnieniowymi).

Niezmiennik 1. Zbiór zmiennych $\{x_1, \ldots, x_{n+m}\}$ dzieli się na dwa rozłączne zbiory: m zmiennych *bazowych* i n zmiennych *niebazowych*. Oznaczmy zbiór indeksów zmiennych bazowych przez $B = \{B_1, \ldots, B_m\}$ (baza) i zmiennych niebazowych przez $N = \{N_1, \ldots, N_n\}$. Program zawiera:

- równanie postaci $z = v + \sum_{j=1}^{n} c_j x_{N_j}$;
- dla każdego $i = 1, \ldots, m$ równanie postaci $x_{B_i} = b_i + \sum_{j=1}^{n} a_{i,j} x_{N_j}$, gdzie $b_i \geq 0$;
- dla każdego $i = 1, \ldots, n+m$ nierówność $x_i \geq 0$,

gdzie v, c_j, b_j, $a_{i,j}$ są stałymi.

W rozważanym przez nas przykładzie, powyższy niezmiennik jest spełniony dla $N = \{1, 2, 3\}$ oraz $B = \{4, 5, 6\}$.

Fakt 1. Jeśli spełniony jest niezmiennik 1, to rozwiązanie $(x_1, \ldots, x_{n+m})$ postaci

$$x_i = \begin{cases} 0 & \text{gdy } i \in N \\ b_j & \text{gdy } i = B_j \text{ dla pewnego } j = 1, \ldots, n \end{cases}$$

jest bazowym rozwiązaniem dopuszczalnym o wartości funkcji celu v.

Proof. Łatwo sprawdzić, że tak zdefiniowane rozwiązanie jest rozwiązaniem dopuszczalnym o wartości funkcji celu v. Aby pokazać, że jest to bazowe rozwiązanie dopuszczalne musimy wskazać $n + m$ liniowo niezależnych ograniczeń spełnionych z równością. W tym celu, dla każdego $i \in B$ wybieramy (jedyną) równość zawierającą x_i oraz dla każdego $i \in N$ nierówność $x_i \geq 0$. □

Z powyższego faktu wynika, że o ile spełniony jest niezmiennik, to faktycznie znajdujemy się w wierzchołku aktualnego progamu w postaci dopełnieniowej. Łatwo sprawdzić też, że po zignorowaniu zmiennych dopełnieniowych otrzymamy wierzchołek odpowiadającego programu w postaci standardowej, a więc faktycznie algorytm będzie generował wierzchołki wielościanu oryginalnego programu liniowego.

Wróćmy do algorytmu simplex. Naszym celem jest zwiększenie zmiennej z. W tym celu spójrzmy na dowolną zmienną niebazową z dodatnim współczynnikiem w funkcji celu. W

tym przypadku możemy wybrać dowolną zmienną z x_1, x_2, x_3 — wybierzmy x_1. Oczywiście powiększając x_1 powiększamy z. Jak bardzo możemy powiększyć x_1, zachowując wszystkie ograniczenia? Dopóki zmienne bazowe pozostają nieujemne, a więc:

$$x_1 := \min\{\tfrac{30}{1}, \tfrac{24}{2}, \tfrac{36}{4}\} = \tfrac{36}{4} = 9.$$

Po takiej operacji przynajmniej jedna zmienna bazowa (w tym przypadku x_6) przyjmuje wartość 0. To pozwala na zmianę bazy (a w konsekwencji zmianę wierzchołka, w którym jesteśmy): zmienna x_1 wchodzi do bazy (jest *zmienną wchodzącą*), a zmienna x_6 wychodzi z bazy (*zmienna wychodząca*). Operacja wymiany bazy (ang. **pivot**) przebiega w dwóch krokach:

1. Rozwiąż równanie zawierające zmienną wychodzącą ze względu na zmienną wchodzącą. W tym przypadku otrzymujemy:
$$x_1 = 9 - \tfrac{1}{4}x_2 - \tfrac{1}{2}x_3 - \tfrac{1}{4}x_6.$$
2. wstaw wynik zamiast x_1 z prawej strony wszystkich równań (czyli uaktualnij współczynniki przy zmiennych niebazowych i wyrazy wolne). W tym przypadku otrzymujemy:
$$\begin{array}{rcrcrcrcr} z &=& 27 &+& \tfrac{1}{4}x_2 &+& \tfrac{1}{2}x_3 &-& \tfrac{3}{4}x_6 \\ x_1 &=& 9 &-& \tfrac{1}{4}x_2 &-& \tfrac{1}{2}x_3 &-& \tfrac{1}{4}x_6 \\ x_4 &=& 21 &-& \tfrac{3}{4}x_2 &-& \tfrac{5}{2}x_3 &+& \tfrac{1}{4}x_6 \\ x_5 &=& 6 &-& \tfrac{3}{2}x_2 &-& 4x_3 &+& \tfrac{1}{2}x_6. \end{array}$$

Fakt 2. Po operacji wymiany bazy otrzymujemy program liniowy o tym samym zbiorze rozwiązań dopuszczalnych.

Otrzymaliśmy rozwiązanie $(9, 0, 0, 21, 6, 0)$ o wartości funkcji celu 27. Wykonajmy kolejną operację wymiany bazy. Teraz jako zmienną wchodzącą możemy wybrać już tylko jedną z dwóch: x_2 lub x_3, bo tylko te zmienne mają dodatnie współczynniki w funkcji celu. Wybierzmy x_3. Podobnie jak poprzednio:

$$x_3 := \min\left\{\frac{9}{\frac{1}{2}}, \frac{21}{\frac{5}{2}}, \frac{6}{4}\right\} = \frac{6}{4} = \frac{3}{2}.$$

A więc x_5 wychodzi z bazy. Otrzymujemy nowy program:

$$\begin{array}{rcrcrcrcr} z &=& \frac{111}{4} &+& \frac{1}{16}x_2 &-& \frac{1}{8}x_5 &-& \frac{11}{16}x_6 \\ x_1 &=& \frac{33}{4} &-& \frac{1}{16}x_2 &+& \frac{1}{8}x_5 &-& \frac{5}{16}x_6 \\ x_3 &=& \frac{3}{2} &-& \frac{3}{8}x_2 &-& \frac{1}{4}x_5 &+& \frac{1}{8}x_6 \\ x_4 &=& \frac{69}{4} &+& \frac{3}{16}x_2 &+& \frac{5}{8}x_5 &-& \frac{1}{16}x_6. \end{array}$$

Teraz jedynym kandydatem do wejścia do bazy jest x_2. Zauważmy, że w ostatnim równaniu współczynnik przed x_2 jest dodatni. Oznacza to, że zwiększając x_2 możemy też zwiększać x_4 zachowując ostatnią równość zawsze spełnioną. A więc przy wyborze zmiennej wychodzącej nie bierzemy pod uwagę x_4:

$$x_2 := \min\left\{\frac{\frac{33}{4}}{\frac{1}{16}}, \frac{\frac{3}{2}}{\frac{3}{8}}\right\} = \frac{\frac{3}{2}}{\frac{3}{8}} = 4.$$

Uwaga. Zastanówmy się jednak przez chwilę, co by było, gdybyśmy nie mieli czego wziąć do minimum, tzn. gdyby istniała zmienna z dodatnim współczynnikiem w funkcji celu i nieujemnymi współczynnikami w pozostałych równaniach? Wtedy powiększając tę zmienną moglibyśmy otrzymać rozwiązanie dopuszczalne o dowolnie wysokiej wartości fukcji celu. W takiej sytuacji algorytm simplex zwraca komunikat "PROGRAM NIEOGRANICZONY" i kończy działanie.

Wracając do naszego programu liniowego, otrzymujemy:

$$\begin{array}{rcrcrcrcl} z & = & 28 & - & \frac{1}{6}x_3 & - & \frac{1}{6}x_5 & - & \frac{2}{3}x_6 \\ x_1 & = & 8 & + & \frac{1}{6}x_3 & + & \frac{1}{6}x_5 & - & \frac{1}{3}x_6 \\ x_2 & = & 4 & - & \frac{8}{3}x_3 & - & \frac{2}{3}x_5 & + & \frac{1}{3}x_6 \\ x_4 & = & 18 & - & \frac{1}{2}x_3 & + & \frac{1}{2}x_5 & + & 0x_6. \end{array}$$

Otrzymaliśmy więc sytuację, gdy nie możemy wykonać operacji wymiany bazy ponieważ wszystkie współczynniki w funkcji celu są ujemne. Jest jednak jasne, że musieliśmy w ten sposób dostać rozwiązanie optymalne programu, ponieważ dla dowolnych *nieujemnych* wartości zmiennych wartość funkcji celu naszego programu nie może przekroczyć aktualnej, czyli 28. Ponieważ w kolejnych krokach algorytmu otrzymywaliśmy równoważne programy liniowe o tym samym zbiorze rozwiązań dopuszczalnych, jest to także rozwiązanie optymalne oryginalnego programu. Odnotujmy te rozważania jako fakt:

Fakt 3. Jeśli w pewnym kroku algorytmu simplex wszystkie współczynniki (przy zmiennych niebazowych) w funkcji celu są ujemne, to znalezione bazowe rozwiązanie dopuszczalne jest optymalnym rozwiązaniem oryginalnego programu.

W naszym przypadku dostaliśmy rozwiązanie $(x_1, x_2, x_3) = (8, 4, 0)$ o wartości funkcji celu 28.

W tej chwili zasada działania algorytmu simplex i jego częściowa poprawność (tzn. poprawność pod warunkiem zatrzymania się programu) powinny być już jasne. Zajmijmy się jeszcze przez chwilę właśnie kwestią warunku stopu i złożoności obliczeniowej.

3.1.1 Pseudokod

Podsumujmy teraz powyższe rozważania podając pseudokod algorytmu Simplex. Stosujemy oznacznia takie jak w niezmienniku 1

1. Sprowadź PL do postaci dopełnieniowej.
2. Znajdź równoważny PL taki, żeby spełniony był niezmiennik 1.
3. Dopóki istnieje $j \in \{1, \ldots, n\}$ takie, że $c_j > 0$ ($c_j =$ współczynnik przed x_{N_j} w aktualnej funkcji celu),
   - 3.1. Wybierz takie j (x_{N_j} jest zmienną wchodzącą).
   - 3.2. Jeśli dla każdego $i = 1, \ldots, m$, jest $a_{i,j} \geq 0$ (tzn. dla każdego równania współczynnik przed x_{N_j} jest nieujemny) zwróć „PROGRAM NIEOGRANICZONY".

3.3. wpp., wybierz i takie, że $\frac{b_i}{-a_{i,j}} = \min\{\frac{b_i}{-a_{i,j}} \mid a_{i,j} < 0\}$ (x_{B_i} jest zmienną wychodzącą).

3.4. wykonaj operację Pivot (j,i)

4. Zwróć rozwiązanie postaci: dla każdego $i = 1, \ldots, n$,

$$x_i = \begin{cases} 0 & \text{gdy } i \in N \\ b_j & \text{gdy } i = B_j \text{ dla pewnego } j = 1, \ldots, m. \end{cases}$$

Podamy teraz pseudokod operacji Pivot(in,out), realizującej usunięcie z bazy $x_{B_{out}}$ i dodanie do niej $x_{N_{in}}$. Przy implementacji, wygodnie jest przechowywać wszystkie stałe w jednej tablicy $a_{i,j}$, gdzie $i \in \{0, \ldots, m\}$, $j \in \{0, \ldots, n\}$, oraz przyjmujemy że $a_{0,0} = v$, $a_{0,j} = c_j$ dla $j = 1, \ldots, n$ oraz $a_{i,0} = b_i$ dla $i = 1, \ldots, m$ (wartości $a_{i,j}$ dla $i, j \geq 1$ mają takie same znaczenie jak wcześniej).

1: $\alpha := a_{out,in}$
2: **for** $j = 0, \ldots, n$ **do**
3: $\quad a_{out,j} := a_{out,j}/(-\alpha)$
4: $a_{out,in} := 1/\alpha$
5: **for** $i \in \{1, \ldots, m\} \setminus \{out\}$ **do**
6: $\quad \beta := a_{i,in}$
7: $\quad a_{i,in} := 0$
8: $\quad$ **for** $j = 0, \ldots, n$ **do**
9: $\quad\quad a_{i,j} := a_{i,j} + \beta \cdot a_{out,j}$
10: $B_{out} :=: N_{in}$

3.1.2 Warunek stopu i reguła Blanda

Jest jasne, że jeśli przy kolejnych operacjach wymiany bazy zawsze otrzymujemy większą (lub, w przypadku minimalizacji, mniejszą) wartość funkcji celu to algorytm musi się zakończyć — po prostu dlatego, że jest ograniczona liczba wierzchołków wielościanu. Poprzednik tej implikacji niestety nie zawsze jest jednak spełniony. Dla przykładu, rozważmy następujący program:

$$\begin{array}{rcccccccc} z &=& 4 &+& 2x_1 &-& x_2 &-& 4x_4 \\ x_3 &=& \frac{1}{2} & & & & & - & \frac{1}{2}x_4 \\ x_5 &=& & - & 2x_1 &+& 4x_2 &+& 3x_4 \\ x_6 &=& & + & x_1 &-& 3x_2 &+& 2x_4. \end{array}$$

Mamy $B = \{3, 5, 6\}$, $N = \{1, 2, 4\}$, $\mathbf{x} = (0, 0, \frac{1}{2}, 0, 0, 0)$ oraz funkcja celu ma wartość $z = 4$. Jako zmienną wchodzącą możemy wybrać jedynie x_1, natomiast jako zmienną wychodzącą jedynie x_5. Po wymianie bazy otrzymujemy:

$$\begin{array}{rcccccccc} z &=& 4 &+& 3x_2 &-& x_4 &-& x_5 \\ x_1 &=& & + & 2x_2 &+& \frac{3}{2}x_4 &-& \frac{1}{2}x_5 \\ x_3 &=& \frac{1}{2} & & & - & \frac{1}{2}x_4 & & \\ x_6 &=& & - & x_2 &+& \frac{7}{2}x_4 &-& \frac{1}{2}x_5. \end{array}$$

Mamy $B = \{1, 3, 6\}$, $N = \{2, 4, 5\}$, $\mathbf{x} = (0, 0, \frac{1}{2}, 0, 0, 0)$ oraz funkcja celu ma wartość $z = 4$. Widzimy, że chociaż zmieniła się baza, bazowe rozwiązanie dopuszczalne pozostało to samo (w szczególności wartość funkcji celu się nie zmieniła). Może być to powodem poważnych kłopotów, a nawet zapętlenia się algorytmu. Początkowo ten problem ignorowano (!), gdyż w praktycznych zastosowaniach pojawia się on niezwykle rzadko. W 1977 (czyli w 30 lat od powstania algorytmu simplex) Robert Bland zaproponował niezwykle prostą heurystykę, o której można pokazać (dowód nie jest bardzo trudny, lecz pominiemy go tutaj), że gwarantuje zakończenie algorytmu.

Twierdzenie 2 (Reguła Blanda)**.** *Jeśli podczas wymiany bazy:*

- *spośród możliwych zmiennych wchodzących wybierana jest zmienna o najmniejszym indeksie oraz,*
- *spośród możliwych zmiennych wychodzących wybierana jest zmienna o najmniejszym indeksie*[8]

to algorytm simplex kończy swoje działanie.

Zapętlenie się algorytmu simplex jest równoważne powrotowi do tej samej bazy. Ponieważ dla programu o m zmiennych bazowych i n zmiennych niebazowych mamy $O(\binom{n+m}{n})$ możliwych baz, więc algorytm simplex z regułą Blanda wykonuje $O(\binom{n+m}{n})$ operacji wymiany bazy (przy każdej z nich wykonuje się $O(nm)$ operacji arytmetycznych). Z drugiej strony, odnotujmy, że

Fakt 4. Istnieją przykłady programów liniowych, dla których algorytm simplex działa w czasie $\Omega(2^n)$.

Mimo to, następujący problem pozostaje otwarty.

Problem 1. *Czy istnieją reguły wyboru zmiennej wchodzącej i wychodzącej, dla których algorytm simplex działa w czasie wielomianowym?*

Jak na razie, najlepsze co udało się uzyskać, to reguły działające w *oczekiwanym* czasie $2^{\tilde{O}(\sqrt{n})}$ [Kalai STOC'92, Math. Prog. 1997, Matousek, Sharir, Welzl Algorithmica 1996].

3.1.3 Znajdowanie początkowego bazowego rozwiązania dopuszczalnego

Do wyjaśnienia pozostała jeszcze jedna kwestia. Zakładaliśmy, że program jest postaci

$$\max \mathbf{c}^T\mathbf{x}, \mathbf{A}\mathbf{x} \leq b, \mathbf{x} \geq 0,$$

oraz że $\mathbf{b} \geq \mathbf{0}$. Wtedy łatwo jest znaleźć równoważny program w postaci dopełnieniowej, który spełnia niezmiennik 1 (innymi słowy: pierwsze bazowe rozwiązanie dopuszczalne). Teraz opiszemy jak to zrobić w przypadku ogólnym. Rozwiązanie będzie dość zaskakujące: żeby znaleźć pierwsze bazowe rozwiązanie dopuszczalne użyjemy algorytmu simplex.

[8] zauważmy, że może być wiele możliwych zmiennych wychodzących, gdy w wielu równaniach iloraz wyrazu wolnego i współczynnika przy zmiennej wchodzącej jest taki sam

1. Sprowadź program do postaci dopełnieniowej:

$$\begin{array}{rcccccccc} \max & & 0 & + & c_1x_1 & + & \dots & + & c_nx_n \\ x_{n+1} & = & b_1 & + & a_{11}x_1 & + & \dots & + & a_{1n}x_n \\ x_{n+2} & = & b_2 & + & a_{21}x_1 & + & \dots & + & a_{2n}x_n \\ \vdots & & & & & & & & \\ x_{n+m} & = & b_m & + & a_{m1}x_1 & + & \dots & + & a_{mn}x_n \\ x_i & \geq & 0 & & & & & & \end{array} \tag{1}$$

2. Dodaj nową zmienną x_0 i zbuduj nowy program:

$$\begin{array}{rcccccccccc} \min & & x_0 & & & & & & & & \\ x_{n+1} & = & b_1 & + & a_{11}x_1 & + & \dots & + & a_{1n}x_n & + & x_0 \\ x_{n+2} & = & b_2 & + & a_{21}x_1 & + & \dots & + & a_{2n}x_n & + & x_0 \\ \vdots & & \vdots & & & & & & & & \\ x_{n+m} & = & b_m & + & a_{m1}x_1 & + & \dots & + & a_{mn}x_n & + & x_0 \\ x_i & \geq & 0 & & & & & & & & \end{array} \tag{2}$$

3. Przyjmij $N = \{0, \dots, n\}$, oraz $B = \{n+1, \dots, n+m\}$. Powyższy PL prawie spełnia niezmiennik, brakuje „tylko" warunku „$b_i \geq 0$ dla każdego i".

4. Wybierz k takie, że $b_k = \min_i\{b_i\}$. Za pomocą operacji Pivot, usuń z bazy x_k i wprowadź do bazy x_0. Otrzymujemy nowy PL:

$$\begin{array}{rcccccccccc} & \min & & -b_k & - & a_{k1}x_1 & - & \dots & - & a_{kn}x_n & + & x_{n+k} \\ i \neq k \Rightarrow & x_{n+i} & = & b_i - b_k & + & (a_{i1} - a_{k1})x_1 & + & \dots & + & (a_{in} - a_{kn})x_n & + & x_{n+k} \\ & x_0 & = & -b_k & - & a_{k1}x_1 & - & \dots & - & a_{kn}x_n & + & x_{n+k} \\ & x_i & \geq & 0 & & & & & & & & \end{array} \tag{3}$$

5. Zauważmy, że program (3) jest równoważny programowi (2) oraz spełnia niezmiennik 1. Za pomocą algorytmu simplex znajdujemy rozwiązanie optymalne $\mathbf{x}^*$. (Zauważmy, że program (2) jest ograniczony: wartością funkcji celu jest wartość x_0, która jest nieujemna.) W używanym algorytmie simplex dokonujemy jednak małej *modyfikacji w regule wyboru zmiennej wychodzącej*: jeśli x_0 może opuścić bazę, to ją opuszcza.

6. Jeśli otrzymaliśmy $x_0^* > 0$ zwracamy informację „PROGRAM SPRZECZNY". Istotnie, gdyby istniało rozwiązanie dopuszczalne $\mathbf{x}$ programu (1), to $(0, \mathbf{x})$ byłoby rozwiązaniem dopuszczalnym programu (2) o wartości funkcji celu 0.

7. Jeśli otrzymaliśmy $x_0^* = 0$ oraz x_0 jest niebazowa, wystarczy z ostatniego PL wygenerowanego przez algorytm simplex usunąć zmienne x_0. Otrzymujemy wtedy program równoważny programowi (1), który spełnia niezmiennik 1

8. Pozostaje przypadek, gdy otrzymaliśmy $x_0^* = 0$ oraz x_0 jest bazowa. Pokażemy, że jest to niemożliwe. Rozważmy ostatnią operację Pivot. Powiedzmy, że zmienną wchodzącą było x_j, a wychodzącą x_i, dla pewnego $i \neq j$. Przed wykonaniem operacji Pivot zarówno x_i jak i x_0 były bazowe, a więc na podstawie niezmiennika 1 odpowiadały im dwa równania:

$$\begin{aligned} x_0 &= b_0 + \ldots + a_{0j}x_j \\ x_i &= b_i + \ldots + a_{ij}x_j. \end{aligned}$$

Po wykonaniu operacji Pivot, równanie zawierające x_0 ma postać

$$x_0 = b_0 + a_{0j} \cdot \frac{b_i}{-a_{ij}} + \sum_{l \in N} a'_{0l} x_{N_l}.$$

Wiemy jednak, że po tej operacji wyraz wolny w tym równaniu jest równy 0, a więc $\frac{b_i}{-a_{ij}} = \frac{b_0}{-a_{0j}}$. Ponieważ x_i została wybrana jako zmienna wychodząca, więc $\frac{b_i}{-a_{ij}} = \min\{\frac{b_\ell}{-a_{\ell j}} \mid a_{\ell j} < 0\}$. Wówczas także $\frac{b_0}{-a_{0j}} = \min\{\frac{b_\ell}{-a_{\ell j}} \mid a_{\ell j} < 0\}$, czyli zmienna x_0 również była kandydatem do opuszczenia bazy, a więc zgodnie z naszą regułą, musiała ją opuścić i cała rozważana sytuacja nie mogła mieć miejsca.

4 Programowanie całkowitoliczbowe

Program (liniowy) całkowitoliczbowy to program liniowy z dodatkowym wymaganiem, aby wartości wszystkich zmiennych były całkowitoliczbowe. Ten z pozoru niewinny warunek całkowicie zmienia złożoność problemu: większość naturalnych problemów NP-trudnych można bardzo łatwo wyrazić jako liniowe programy całkowitoliczbowe, np. problem pokrycia wierzchołkowego w grafie $G = (V, E)$ jest równoważny programowi:

$$\begin{array}{lll} \min & \sum_{v \in V} x_v & \\ & x_u + x_v \geq 1 & \text{dla każdego } uv \in E \\ & x_v \in \{0, 1\} & \text{dla każdego } v \in V, \end{array} \tag{4}$$

ponieważ ostatni warunek możemy zastąpić przez koniunkcję $0 \leq x_v \leq 1$ i $x_v \in \mathbb{Z}$. Jeśli min zastąpimy przez max, a $\geq$ przez $\leq$ otrzymamy z kolei problem najmniejszego zbioru niezależnego.

Wniosek 3. *Problem programowania liniowego całkowitoliczbowego jest NP-trudny.*

4.1 Unimodularność

Mówimy, że macierz $\mathbf{A}$ jest *całkowicie unimodularna* jeśli dla każdej podmacierzy $\mathbf{A}'$ macierzy $\mathbf{A}$, mamy $\det \mathbf{A}' \in \{-1, 0, 1\}$.

Twierdzenie 3. *Jeśli wektor* $\mathbf{b}$ *jest calkowitoliczbowy i macierz* $\mathbf{A}$ *jest całkowicie unimodularna to program* $\mathbf{Ax} \leq \mathbf{b}$ *ma wierzchołki całkowitoliczbowe.*

Proof. Ćwiczenie. Wskazówka: skorzystaj z tego, że wierzchołki są bazowymi rozwiązaniami dopuszczalnymi oraz ze wzorów Cramera. □

Zauważmy, że jeśli mamy pewien problem który jest równoważny liniowemu programowi całkowitoczbowemu (I), o całkowitych wyrazach wolnych, oraz jeśli pokażemy, że macierz tego programu jest unimodularna, to możemy:

1. zignorować warunek całkowitoliczbowości otrzymując program liniowy (L),
2. rozwiązać (L) w czasie wielomianowym (np. metodą elipsoidalną) otrzymując rozwiązanie optymalne $\mathbf{x}^*$,
3. znaleźć wierzchołek $\mathbf{x}'$ o wartości funkcji celu takiej samej jak dla $\mathbf{x}^*$ (wielomianowo, np. tak jak w dowodzie twierdzenia 1)
4. zwrócić $\mathbf{x}'$ jako rozwiązanie programu całkowitoliczbowego (I).

Unimodularność jest użytecznym narzędziem w dowodzeniu, że dany problem leży w klasie $\mathbf{P}$.

5 Dualność

5.1 Motywacja: certyfikat optymalności

Dla niektórych problemów znane są algorytmy, które wraz z rozwiązaniem generują tzw. certyfikat poprawności, czyli dodatkową informację z pomocą której możemy łatwo (tzn. algorytmem, który jest nie tylko wielomianowy, ale również istotnie prostszy od algorytmu generującego rozwiązanie) sprawdzić, czy zwrócone rozwiązanie jest poprawne (lub optymalne w przypadku problemów optymalizacyjnych). Algorytmy te określane są mianem *algorytmów certyfikujących*. Oto kilka przykładów. Algorytm sprawdzający, czy dany graf jest dwudzielny może zwracać 2-kolorowanie takiego grafu lub cykl nieparzysty. Algorytm znajdowania maksymalnego przepływu wraz z przepływem może zwracać minimalny przekrój. Istnieje algorytm testujący planarność grafów, który zwraca rysunek na płaszczyźnie bez przecięć krawędzi lub podgraf homeomorficzny z $K_{3,3}$ lub K_5. Algorytmy certyfikujące oprócz znaczenia teoretycznego (aby generować takie certyfikaty należy dobrze zrozumieć problem) mają dużą wartość praktyczną: pomyślmy choćby o testowaniu poprawności kodu.

Czy dla programowania liniowego istnieją certyfikaty poprawności? Rozważmy decyzyjną wersję problemu programowania liniowego: mając dany (minimalizacyjny) PL i liczbę δ rozstrzygnąć, czy istnieje dopuszczalny $\mathbf{x}$ taki, że $\mathbf{c}^T\mathbf{x} \leq \delta$. Załóżmy dla uproszczenia, że nasz program jest ograniczony. Wraz z odpowiedzią TAK algorytm certyfikujący może zwrócić współrzędne $\mathbf{x}$ (można nawet pokazać, że te współrzędne można reprezentować w pamięci wielomianowej względem rozmiaru danych). A więc istnieje certyfikat pozytywny. Z punktu widzenia teorii złożoności, dowodzi to, że problem jest w klasie NP. W dodatku algorytm weryfikacji certyfikatu jest banalny: po prostu sprawdzamy odpowiednie nierówności. Czy możemy podać certyfikat

negatywny, tzn. certyfikat poprawności dla odpowiedzi NIE? Gdyby ten certyfikat był również krótki, mielibyśmy dowód, że programowanie liniowe jest w klasie co-NP.

W tym rozdziale przedstawimy pojęcie dualności programowania liniowego, które dostarcza odpowiedzi na powyższe pytania.

5.2 Poszukiwanie górnego ograniczenia

Rozważmy następujący PL w postaci standardowej:

$$\begin{array}{ll} \text{zmaksymalizuj} & 3x_1 - x_2 + 2x_3 \\ \text{z zachowaniem warunków} & x_1 - x_2 + \frac{1}{2}x_3 \leq 4 \\ & 4x_1 + 2x_2 + 3x_3 \leq 20 \\ & x_1, x_2, x_3 \geq 0 \end{array} \tag{5}$$

Łatwo sprawdzić, że punkt $(x_1 = 2, x_2 = 0, x_3 = 4)$ jest rozwiązaniem dopuszczalnym o wartości funkcji celu 14.

Zauważmy, że skoro $x_1, x_2, x_3 \geq 0$, to $3x_1 \leq 4x_1$, a także $-x_2 \leq 2x_2$ oraz $2x_3 \leq 3x_3$. Stąd, $3x_1 - x_2 + 2x_3 \leq 4x_1 + 2x_2 + 3x_3 \leq 20$. Dostaliśmy górne ograniczenie! W tym przypadku możemy zauważyć nawet więcej. Ponieważ $3 \leq 1 + \frac{1}{2} \cdot 4$, $-1 \leq -1 + \frac{1}{2} \cdot 2$ oraz $2 \leq \frac{1}{2} + \frac{1}{2} \cdot 3$, więc

$$\begin{aligned} 3x_1 - x_2 + 2x_3 \leq x_1 - x_2 + \tfrac{1}{2}x_3 + \\ \tfrac{1}{2} \cdot (4x_1 + 2x_2 + 3x_3) \leq 4 + \tfrac{1}{2} \cdot 20 = 14. \end{aligned}$$

Udowodniliśmy (choć, trzeba przyznać, dość fartownie), że wartość funkcji celu nigdy nie przekracza 14, a więc mamy certyfikat optymalności, w dodatku bardzo krotki (kilka nierówności) i prosty do sprawdzenia. Możemy to rozumowanie uogólnić: można brać dowolne kombinacje liniowe nierówności t.ż.:

- współczynniki kombinacji liniowej są nieujemne (bo inaczej odwrócą się kierunki nierówności),
- dla dowolnego i, współczynnik funkcji celu przy x_i jest ograniczony z góry przez odpowiednią kombinację liniową współczynników przy x_i w nierównościach.

Można to zapisać za pomocą programu liniowego:

$$\begin{array}{ll} \text{zminimalizuj} & 4y_1 + 20y_2 \\ \text{z zachowaniem warunków} & y_1 + 4y_2 \geq 3 \\ & -y_1 + 2y_2 \geq -1 \\ & \frac{1}{2}y_1 + 3y_2 \geq 2 \\ & y_1, y_2 \geq 0. \end{array} \tag{6}$$

Powyższy program będziemy nazywać *programem dualnym* do programu (5). Ogólnie, dla programu (będziemy go nazywać programem *prymalnym*)

$$\begin{array}{lll} \text{zmaksymalizuj} & \sum_{j=1}^{n} c_j x_j & \\ \text{z zachowaniem warunków} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1, \ldots, m \\ & x_j \geq 0 & j = 1, \ldots, n \end{array} \tag{7}$$

program dualny ma postać:

$$\begin{array}{lll} \text{zminimalizuj} & \sum_{i=1}^{m} b_i y_i & \\ \text{z zachowaniem warunków} & \sum_{i=1}^{m} a_{ij} y_i \geq c_j & j = 1, \ldots, n \\ & y_i \geq 0 & i = 1, \ldots, m. \end{array} \tag{8}$$

Zauważmy, że macierz współczynników ograniczeń programu (8) jest transpozycją macierzy dla programu (7) (porównaj też dla programów (5) i (6)). Daje to niezwykle prosty, mechaniczny sposób konstruowania programu dualnego. Mianowicie dla programu

$$\begin{array}{ll} \text{zmaksymalizuj} & \mathbf{c}^T \mathbf{x} \\ \text{z zachowaniem warunków} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{9}$$

program dualny ma postać:

$$\begin{array}{ll} \text{zminimalizuj} & \mathbf{b}^T \mathbf{y} \\ \text{z zachowaniem warunków} & \mathbf{A}^T \mathbf{y} \geq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0} \end{array} \tag{10}$$

Dualność jest relacją symetryczną, tzn. mówimy także, że (7) jest dualny do (8). Innymi słowy, program dualny do programu dualnego to program prymalny.

5.3 Słaba dualność i komplementarne warunki swobody

Pozostańmy przy programach w postaci standardowej. Z konstrukcji programu dualnego wynika następujący fakt (dla porządku podamy jednak dowód).

Twierdzenie 4 (słaba dualność). *Niech* $\mathbf{x}$ *i* $\mathbf{y}$ *będą dowolnymi rozwiązaniami dopuszczalnymi odpowiednio programów* (7) *i* (8)*. Wówczas* $\mathbf{c}^T \mathbf{x} \leq \mathbf{b}^T \mathbf{y}$.

Proof. Ponieważ dla każdego $j = 1, \ldots, n$, mamy $x_j \geq 0$ oraz $\sum_{i=1}^{m} a_{ij} y_i \geq c_j$, więc

$$c_j x_j \leq \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j \qquad \forall j = 1, \ldots, n. \tag{11}$$

Podobnie, ponieważ dla każdego $i = 1, \ldots, m$, mamy $y_i \geq 0$ oraz $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$, więc

$$\left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq b_i y_i \qquad \forall i = 1, \ldots, m. \tag{12}$$

Stąd,

$$\mathbf{c}^T\mathbf{x} = \sum_{j=1}^{n} c_j x_j \leq \sum_{j=1}^{n} \left(\sum_{i=1}^{m} a_{ij} y_i \right) x_j = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j \right) y_i \leq \sum_{i=1}^{m} b_i y_i = \mathbf{b}^T\mathbf{y}. \tag{13}$$

□

Zastanówmy się, kiedy rozwiązania optymalne programu prymalnego (7) i dualnego (8) spotykają się, czyli $\mathbf{c}^T\mathbf{x} = \mathbf{b}^T\mathbf{y}$. Z dowodu twierdzenia 4 widzimy, że jest tak wtedy i tylko wtedy gdy obie nierówności w (13) są równościami. Tak może się wydarzyć tylko wtedy, gdy gdy wszystkie nierówności w (11) i (12) są równościami. To dowodzi następującego twierdzenia:

Twierdzenie 5. *Niech* $\mathbf{x}$ *i* $\mathbf{y}$ *będą rozwiązaniami dopuszczalnymi odpowiednio dla zadania prymalnego i dualnego w postaci standardowej. Rozwiązania* $\mathbf{x}$ *i* $\mathbf{y}$ *są oba optymalne wtedy i tylko wtedy gdy*

(i) **prymalne komplementarne warunki swobody**
dla każdego $j = 1, \ldots, n$ *albo* $x_j = 0$ *albo* $\sum_{i=1}^{m} a_{ij} y_i = c_j$.
(albo $x_j = 0$ *albo* j*-ta nierówność programu dualnego jest spełniona z równością.)*

(ii) **dualne komplementarne warunki swobody**
dla każdego $i = 1, \ldots, m$ *albo* $y_i = 0$ *albo* $\sum_{j=1}^{n} a_{ij} x_j = b_i$.
(albo $y_i = 0$ *albo* i*-ta nierówność programu prymalnego jest spełniona z równością.)*

5.4 Programy dualne do ogólnych programów liniowych

Nawet gdy program nie jest w postaci standardowej możemy napisać program dualny, kierując się tą samą zasadą: poszukujemy jak najlepszego górnego ograniczenia na wartość funkcji celu. Zobaczmy np. co się dzieje, gdy program zawiera równość:

$$\begin{array}{ll} \text{zmaksymalizuj} & 3x_1 - x_2 + 2x_3 \\ \text{z zachowaniem warunków} & x_1 - x_2 + \frac{1}{2}x_3 \leq 4 \\ & -4x_1 - 2x_2 - 3x_3 = -20 \\ & x_1, x_2, x_3 \geq 0 \end{array} \tag{14}$$

Podobnie jak poprzednio, dodając pierwszy warunek pomnożony przez $y_1 = 1$ do drugiego warunku pomnożonego przez $y_2 = -\frac{1}{2}$ dostajemy ograniczenie górne równe 14. Zauważmy, że w kombinacji liniowej warunków, współczynniki dla nierówności muszą być nieujemne, natomiast dla równości mogą być dowolne (w tym przypadku wybraliśmy ujemny). Program dualny wygląda następująco:

$$\begin{array}{ll} \text{zminimalizuj} & 4y_1 + 20y_2 \\ \text{z zachowaniem warunków} & y_1 + 4y_2 \geq 3 \\ & -y_1 + 2y_2 \geq -1 \\ & \frac{1}{2}y_1 + 3y_2 \geq 2 \\ & y_1 \geq 0. \end{array} \tag{15}$$

A co by się stało, gdyby w programie prymalnym dodatkowo mogły się pojawiać zmienne bez warunku nieujemności? Rozważmy np.

$$\begin{array}{ll} \text{zmaksymalizuj} & 3x_1 - x_2 + 2x_3 \\ \text{z zachowaniem warunków} & x_1 - x_2 + \frac{1}{2}x_3 \leq 4 \\ & 4x_1 + 2x_2 + 3x_3 = 20 \\ & x_1, x_3 \geq 0 \end{array} \tag{16}$$

Wówczas nie możemy już napisać, że $3x_1 - x_2 + 2x_3 \leq 4x_1 + 2x_2 + 3x_3$, gdyż niekoniecznie $-x_2 \leq 2x_2$. Jeśli jednak pomnożymy pierwszą nierówność przez 3 i dodamy do drugiej, dostaniemy:

$$\begin{aligned} 3x_1 - x_2 + 2x_3 \leq 3(x_1 - x_2 + \tfrac{1}{2}x_3) + \\ 4x_1 + 2x_2 + 3x_3 \leq 12 + 20 = 32, \end{aligned}$$

gdyż $3x_1 \leq 7x_1$, $-x_2 = -x_2$ oraz $2x_3 \leq 3x_3$. Znalezienie najlepszej kombinacji liniowej warunków odpowiada programowi:

$$\begin{array}{ll} \text{zminimalizuj} & 4y_1 + 20y_2 \\ \text{z zachowaniem warunków} & y_1 + 4y_2 \geq 3 \\ & -y_1 + 2y_2 = -1 \\ & \frac{1}{2}y_1 + 3y_2 \geq 2 \\ & y_1 \geq 0. \end{array} \tag{17}$$

Ogólnie, program dualny konstruujemy zgodnie z poniższą zasadą:

	PRYMALNY	$\leftrightarrow$	DUALNY	
f. celu	$\max \mathbf{c}^T\mathbf{x}$	$\leftrightarrow$	f.celu	$\min \mathbf{b}^T\mathbf{y}$
i-ty warunek	$\sum_{j=1}^{n} a_{ij}x_j \leq b_i$	$\leftrightarrow$	i-ta zmienna	$y_i \geq 0$
i-ty warunek	$\sum_{j=1}^{n} a_{ij}x_j = b_i$	$\leftrightarrow$	i-ta zmienna	y_i nieograniczone.
j-ta zmienna	$x_j \geq 0$	$\leftrightarrow$	j-ty warunek	$\sum_{i=1}^{m} a_{ij} \geq c_j$
j-ta zmienna	x_j nieograniczone	$\leftrightarrow$	j-ty warunek	$\sum_{i=1}^{m} a_{ij} = c_j$

Można łatwo sprawdzić, że program dualny zbudowany zgodnie z powyższymi wytycznymi również spełnia twierdzenie o słabej dualności (a także twierdzenie o silnej dualności, które udowodnimy w kolejnym punkcie).

5.5 Silna dualność

Twierdzenie o słabej dualności można również wyrazić w następujący sposób:

Wniosek 4 (słaba dualność). *Jeśli z jest wartością funkcji celu rozwiązania optymalnego prymalnego minimalizacyjnego PL, natomiast w jest wartością funkcji celu rozwiązania optymalnego programu dualnego, to $z \geq w$.*

Wynika stąd w szczególności, że gdy $z = w$, to rozwiązanie optymalne programu dualnego jest certyfikatem optymalności naszego rozwiązania programu prymalnego. Okazuje się, że tak jest *zawsze*!

Twierdzenie 6 (silna dualność). *Jeśli z jest wartością funkcji celu rozwiązania optymalnego prymalnego PL, natomiast w jest wartością funkcji celu rozwiązania optymalnego programu dualnego, to $z = w$.*

Proof. Dla uproszczenia przeprowadzimy dowód dla przypadku programu w maksymalizacyjnej postaci standardowej (dowód dla ogólnej postaci jest analogiczny). Oto program prymalny:

$$\begin{array}{ll} \text{zmaksymalizuj} & \mathbf{c}^T\mathbf{x} \\ \text{z zachowaniem warunków} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}, \end{array} \tag{18}$$

oraz program dualny:

$$\begin{array}{ll} \text{zminimalizuj} & \mathbf{b}^T\mathbf{y} \\ \text{z zachowaniem warunków} & \mathbf{A}^T\mathbf{y} \geq \mathbf{c} \\ & \mathbf{y} \geq \mathbf{0}. \end{array} \tag{19}$$

Niech $\mathbf{x}^*$ będzie rozwiązaniem optymalnym programu prymalnego (18). Ze słabej dualności, wystarczy pokazać rozwiązanie dopuszczalne $\mathbf{y}$ programu dualnego (19) t.ż. $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}$. Uruchamiamy alorytm simplex na programie (18). W pierwszej iteracji algorytm simplex buduje program

$$\begin{array}{lll} \text{zmaksymalizuj} & z & \\ \text{z zachowaniem warunków} & z = \sum_{j=1}^{n} c_j x_j & \\ & x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij} x_j & i = 1, \ldots, m \\ & x_j \geq 0 & j = 1, \ldots, n+m. \end{array} \tag{20}$$

Rozważmy program z ostatniej iteracji algorytmu simplex.

$$\begin{array}{lll} \text{zmaksymalizuj} & z & \\ \text{z zachowaniem warunków} & z = v + \sum_{j \in N} c'_j x_j & \\ & x_i = b'_i - \sum_{j \in N} a'_{ij} x_j & i \in B \\ & x_j \geq 0 & j = 1, \ldots, n+m. \end{array} \tag{21}$$

Ponieważ program prymalny jest ograniczony, więc dla każdego $j \in N$, $c'_j \leq 0$ oraz algorytm simplex zwraca rozwiązanie optymalne $\mathbf{x}^*$ takie, że

$$x_i^* = \begin{cases} 0 & \text{gdy } i \in N \\ b'_i & \text{gdy } i \in B. \end{cases} \tag{22}$$

.

Określimy teraz pewne rozwiązanie programu dualnego (19):

$$y_i := \begin{cases} -c'_{n+i} & \text{gdy } n+i \in N \\ 0 & \text{w p. p.} \end{cases} \tag{23}$$

.

Pokażemy teraz, że $\mathbf{y}$ jest dopuszczalny o wartości funkcji celu równej $\mathbf{c}^T\mathbf{x}^*$. Określamy $c'_j := 0$ dla $j \in B$. Wówczas pierwszy warunek programu (21) możemy zapisać jako $z = v + \sum_{j=1}^{n+m} c'_j x_j$.

Weźmy **dowolne** $(x_1, \ldots, x_n) \in \mathbb{R}^n_{\geq 0}$. Określmy $x_{n+i} = b_i - \sum_{j=1}^{n} a_{ij}x_j$ oraz $z = \sum_{j=1}^{n} c_j x_j$. Wówczas $(z, x_1, \ldots, x_{n+m})$ jest rozwiązaniem dopuszczalnym programu (20). Ponieważ program (21) jest równoważny (20) (tzn. powstaje z (20) na drodze ciągu operacji elementarnych), więc $(z, x_1, \ldots, x_{n+m})$ jest również rozwiązaniem dopuszczalnym programu (21). Z faktu, iż w obu programach spełnione są warunki zawierające zmienną z mamy, iż:

$$\begin{aligned} \sum_{j=1}^{n} c_j x_j &= v + \sum_{j=1}^{n+m} c'_j x_j \\ &= v + \sum_{j=1}^{n} c'_j x_j + \sum_{i=1}^{m} c'_{n+i} x_{n+i} \\ &= v + \sum_{j=1}^{n} c'_j x_j + \sum_{i=1}^{m} (-y_i)(b_i - \sum_{j=1}^{n} a_{ij} x_j). \end{aligned}$$

Po przegrupowaniu dostajemy, że dla dowolnego $(x_1, \ldots, x_n) \in \mathbb{R}^n_{\geq 0}$,

$$\sum_{j=1}^{n} c_j x_j = \left(v - \sum_{i=1}^{m} y_i b_i \right) + \sum_{j=1}^{n} \left(c'_j + \sum_{i=1}^{m} y_i a_{ij} \right) x_j. \tag{24}$$

Podstawiamy $(x_1, \ldots, x_n) = (0, \ldots, 0)$ i mamy $v = \sum_{i=1}^{m} y_i b_i$. Ponieważ $\mathbf{c}^T\mathbf{x}^* = v$, więc $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}$. Pozostaje pokazać dopuszczalność $\mathbf{y}$. W tym celu rozważmy dowolne $k \in \{1, \ldots, n\}$. Podstawiamy $x_j = [j = k]$ dla każdego $j = 1, \ldots, n$. Otrzymujemy

$$c_k = \underbrace{(v - \sum_{i=1}^{m} y_i b_i)}_{=0} + \underbrace{c'_k}_{\leq 0} + \sum_{i=1}^{m} y_i a_{ik}, \tag{25}$$

co implikuje $\sum_{i=1}^{m} y_i a_{ik} \geq c_k$ dla każdego k, czyli $\mathbf{A}^T\mathbf{y} \geq \mathbf{c}$. Zauważmy, że ponieważ $c'_j \leq 0$ dla każdego j, więc $\mathbf{y} \geq 0$. Pokazaliśmy zatem, że $\mathbf{y}$ jest rozwiązaniem dopuszczalnym programu (19) o wartości funkcji celu $\mathbf{c}^T\mathbf{x}^*$, co kończy dowód. □

Z powyższego dowodu wynika, że algorytm simplex znajduje równocześnie rozwiązania optymalne programu prymalnego i dualnego. W niektórych sytuacjach możemy uzyskać skrócenie czasu działania algorytmu stosując tzw. dualny algorytm simplex, tzn. uruchamiać algorytm simplex dla programu dualnego.

5.6 Zastosowania dualności

5.6.1 Lemat Farkasa

Przypomnijmy, że dla układów równań prawdziwa jest następująca elegancka własność:

Lemat 7. *Są równoważne:*

(i) istnieje $\mathbf{x}$ *taki, że* $\mathbf{Ax} = \mathbf{b}$*,*

(ii) **nie** *istnieje* $\mathbf{y}$ *taki, że* $\mathbf{y}^T\mathbf{A} = \mathbf{0}$ *oraz* $\mathbf{y}^T\mathbf{b} \neq \mathbf{0}$*.*

Innymi słowy, albo układ ma rozwiązanie, albo istnieje taka kombinacja liniowa $\mathbf{y}$ jego równań, która prowadzi do sprzeczności. Odpowiednik tego faktu dla programów liniowych nosi nazwę Lemat Farkasa.

Lemat 8 (Farkasa). *Są równoważne:*

(i) istnieje $\mathbf{x}$ *taki, że* $\mathbf{Ax} \leq \mathbf{b}$*,*

(ii) nie istnieje $\mathbf{y} \geq \mathbf{0}$ *taki, że* $\mathbf{A}^T\mathbf{y} = \mathbf{0}$ *oraz* $\mathbf{y}^T\mathbf{b} < 0$*.*

Dowód. Rozważmy program liniowy:

$$\begin{array}{ll} \text{zmaksymalizuj} & 0 \\ \text{z zachowaniem warunków} & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \text{ nieograniczony} \end{array} \tag{26}$$

Napiszmy program dualny zgodnie z zasadami z punktu 5.4.

$$\begin{array}{ll} \text{zminimalizuj} & \mathbf{b}^T\mathbf{y} \\ \text{z zachowaniem warunków} & \mathbf{A}^T\mathbf{y} = \mathbf{0} \\ & \mathbf{y} \geq 0. \end{array} \tag{27}$$

(i) $\Rightarrow$ (ii). Jeśli istnieje $\mathbf{x}$ taki, że $\mathbf{Ax} \leq \mathbf{b}$, to program prymalny (26) jest dopuszczalny i ma rozwiązanie optymalne o koszcie 0. Z twierdzenia o (słabej) dualności, program dualny nie ma rozwiązań dopuszczalnych o wartości funkcji celu < 0.

(ii) $\Rightarrow$ (i). Jeśli nie istnieje $\mathbf{y} \geq \mathbf{0}$ taki, że $\mathbf{A}^T\mathbf{y} = \mathbf{0}$ oraz $\mathbf{y}^T\mathbf{b} < 0$, oznacza to, że $\mathbf{y} = \mathbf{0}$ jest rozwiązaniem optymalnym programu dualnego (27). Z twierdzenia o (silnej) dualności program prymalny ma rozwiązanie optymalne, a więc układ nierówności $\mathbf{Ax} \leq \mathbf{b}$ jest niesprzeczny. □

Co ciekawe, twierdzenie o silnej dualności można łatwo otrzymać jako wniosek z lematu Farkasa (który również można udowodnić niezależnie, na gruncie geometrii i algebry liniowej). Poniżej pokazujemy dowód twierdzenia o dualności (dla urozmaicenia w wersji dla postaci dopełnieniowej).

Twierdzenie 7 (silna dualność). *Jeśli* z *jest wartością funkcji celu* $\mathbf{c}$ *rozwiązania optymalnego prymalnego minimalizacyjnego PL w postaci dopełnieniowej* $\mathbf{A}^T\mathbf{x} = \mathbf{b}$*,* $\mathbf{x} \geq \mathbf{0}$*, natomiast* w *jest wartością funkcji celu rozwiązania optymalnego programu dualnego, to* $z = w$*.*

Dowód. Ze słabej dualności, $z \geq w$. Załóżmy, że $z > w$. Wtedy $\{\mathbf{y} \mid \mathbf{A}^T\mathbf{y} \leq \mathbf{c}, \mathbf{b}^T\mathbf{y} \geq z\} = \emptyset$. Z lematu Farkasa mamy, że

$$\text{istnieje } \begin{bmatrix} \mathbf{x} \\ q \end{bmatrix} \geq \mathbf{0} \text{ takie, że } [\mathbf{A} \mid -\mathbf{b}] \begin{bmatrix} \mathbf{x} \\ q \end{bmatrix} = \mathbf{0} \text{ oraz } [\mathbf{x} \mid q] \begin{bmatrix} \mathbf{c} \\ -z \end{bmatrix} < \mathbf{0}.$$

Stąd $x \geq \mathbf{0}$, $q \geq 0$, $\mathbf{Ax} = \mathbf{b}q$ oraz $\mathbf{c}^T\mathbf{x} < q\mathbf{z}$. Rozważamy dwa przypadki w zależności od q.

Jeśli $q > 0$, to mamy $\mathbf{A}\frac{\mathbf{x}}{q} = \mathbf{b}$, $\frac{\mathbf{x}}{q} \geq \mathbf{0}$ oraz $\mathbf{c}^T\frac{\mathbf{x}}{q} < z$ czyli $\frac{\mathbf{x}}{q}$ jest rozwiązaniem dopuszczalnym o mniejszej wartości funkcji celu niż z, sprzeczność.

Jeśli $q = 0$, to $\mathbf{Ax} = \mathbf{0}$ i $\mathbf{c}^T\mathbf{x} < \mathbf{0}$. Niech $\mathbf{x}^*$ będzie rozwiązaniem dopuszczalnym takim, że $\mathbf{c}^T\mathbf{x}^* = z$. Wówczas $\mathbf{A}(\mathbf{x}^* + \mathbf{x}) = \mathbf{b}$ oraz $\mathbf{x}^* + \mathbf{x} \geq \mathbf{0}$ ale $\mathbf{c}^T(\mathbf{x}^* + \mathbf{x}) < \mathbf{c}^T\mathbf{x}^* = z$, ponownie sprzeczność. □

5.6.2 Twierdzenie Königa-Egervary'ego

Wiele z klasyczynych twierdzeń mini-maksowych okazuje się być szczególnymi przypadkami twierdzenia o dualności programów liniowych. Dla przykładu, pokażemy, że jest tak dla twierdzenia Königa-Egervary'ego.

Rozważmy problem maksymalnego skojarzenia w danym grafie $G = (V, E)$. Problem ten możemy łatwo sformułować jako zadanie programowania liniowego całkowitoliczbowego.

$$\begin{array}{lll} \max & \sum_{e \in E} x_e & \\ & \sum_{vw \in E} x_{vw} \leq 1 & \text{dla każdego } v \in V \\ & x_e \in \{0, 1\} & \text{dla każdego } e \in E. \end{array} \tag{28}$$

Relaksacja tego programu wygląda następująco:

$$\begin{array}{lll} \max & \sum_{e \in E} x_e & \\ & \sum_{vw \in E} x_{vw} \leq 1 & \text{dla każdego } v \in V \\ & x_e \geq 0 & \text{dla każdego } e \in E. \end{array} \tag{29}$$

Zauważmy, że w powyższym programie mogliśmy opuścić warunek $x_e \leq 1$, gdyż i tak jest on spełniony dla każdego rozwiązania dopuszczalnego programu (29).

Twierdzenie 8. *Jeśli G jest dwudzielny, to macierz programu* (29) *jest unimodularna.*

Dowód. Ćwiczenie. □

Rozważmy program dualny:

$$\begin{array}{lll} \min & \sum_{v \in V} y_v & \\ & y_u + y_v \geq 1 & \text{dla każdego } uv \in E \\ & y_v \geq 0 & \text{dla każdego } v \in V. \end{array} \tag{30}$$

W powyższym programie moglibyśmy dodać warunek $y_v \leq 1$: i tak jest on spełniony dla każdego rozwiązania optymalnego programu (30). Ponieważ macierz programu (30) jest transpozycją macierzy programu (29), więc również jest unimodularna. Stąd program (30)

ma rozwiązanie optymalne, które jest również rozwiązaniem optymalnym programu całkowitoliczbowego

$$\begin{array}{ll} \min & \sum_{v\in V} y_v \\ & y_u + y_v \geq 1 \quad \text{dla każdego } uv \in E \\ & y_v \in \{0,1\} \quad \text{dla każdego } v \in V. \end{array} \tag{31}$$

Z kolei zauważamy, że powyższy program jest równoważny problemowi znalezienia najmniejszego pokrycia wierzchołkowego grafu G. Na podstawie twierdzenia o silnej dualności otrzymujemy

Twierdzenie 9 (König, Egervary)**.** *W grafie dwudzielnym rozmiar największego skojarzenia jest równy liczności najmniejszego pokrycia wierzchołkowego.*

5.6.3 Twierdzenie minimaksowe von Neumanna

Rozważmy następującą (niektórym czytelnikom być może znaną z dzieciństwa?) grę. Mamy dwóch graczy, każdy z nich niezależnie wybiera jedną z trzech opcji: nożyczki, papier lub kamień a następnie równocześnie ujawniają sobie swoje wybory. Jeśli wybrali to samo, gra kończy się remisem. W przeciwnym przypadku wygrywa jeden z graczy zgodnie z zasadą: nożyczki wygrywają z papierem (tną go!), papier wygrywa z kamieniem (bo kamień można zawinąć w papier!), kamień wygrywa z nożyczkami (tępi je!).

Możemy uogólnić tę grę w następujący sposób. Dana jest dowolna (niekoniecznie kwadratowa) macierz $A = [a_{ij}]$ liczb rzeczywistych oraz dwóch graczy: W (gracz wierszowy) i K (gracz kolumnowy). Gracz W wybiera wiersz w macierzy A, natomiast gracz K niezależnie wybiera kolumnę k macierzy A. Następnie równocześnie ujawniają sobie swoje wybory i gracz W dostaje od gracza K a_{wk} złotych. Jest to tzw. *gra o sumie zerowej* (tzn. suma zysków graczy wynosi 0). Dla przykładu, następująca macierz odpowiada grze w nożyczki, papier i kamień (zakładamy, że wiersz/kolumna 1 odpowiada nożyczkom, 2 papierowi, 3 kamieniowi):

$$A = \begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix}. \tag{32}$$

A oto inna gra o sumie zerowej, której będziemy używać jako przykład w dalszej części wykładu.

$$A = \begin{bmatrix} 4 & -1 \\ 1 & 2 \end{bmatrix}. \tag{33}$$

Jak mogą wyglądać strategie w grze tego typu? Powiedzmy, że W wybiera wiersz 2. Taką strategię (wybór konkretnego wiersza/kolumny) nazywamy *czystą*. Taka strategia ma pewną wadę: gdy przeciwnik (np. po wielu rozegranych rozgrywkach) domyśli się, że zawsze wybieramy wiersz 2, zacznie wybierać kolumnę 1: wówczas W zawsze wygrywa tylko 1. Oczywiście wówczas, czyli gdy K wybiera zawsze 1, to W opłaca się zmienić stategię i wybierać wiersz 1. Ale wówczas strategię zmieni również K itd. Innymi słowy, jeśli obaj gracze mają ustaloną jakąś

strategię czystą, to jednemu z nich opłaca się zmienić swoją strategię. (Mówimy wtedy, że nasza gra nie ma equilibrium Nasha w strategiach czystych.)

Rozważymy jednak bardziej wyrafinowane strategie. *Strategią mieszaną* gracza W nazywamy dowolny rozkład prawdopodobieństwa na wierszach macierzy A. Np. gracz W mógłby z prawdopobieństewm $p_1 = \frac{1}{3}$ wybierać wiersz 1 i z prawdopobieństewm $p_2 = \frac{2}{3}$ wybierać wiersz 2. (Analogicznie definiujemy strategię mieszaną dla gracza K). Jak wygląda *optymalna* strategia mieszana dla W? Gracz W maksymalizuje wartość oczekiwaną swojej wygranej. Gracz K ma do wyboru 2 ruchy. Gdy wybierze kolumnę 1, wartość oczekiwana kwoty, którą zapłaci graczowi W wynosi $4{\cdot}p_1{+}1{\cdot}p_2$. Gdy wybierze kolumnę 2, ta wartość wynosi $(-1){\cdot}p_1{+}2{\cdot}p_2$. Gracz K chce zapłacić jak najmniej, a więc wybierze bardziej korzystną opcję i zapłaci $\min\{4p_1{+}p_2, -p_1{+}2p_2\}$. Teoretycznie K mógłby grać przeciwko W również używając strategii mieszanej. Zauważmy jednak że jeśli, tak jak w powyższej analizie, K zna p_1 i p_2 to opłaca mu się grać jedną z 2 strategii czystych, tzn. $\min_{\substack{q_1+q_2=1\\ q_1,q_2\geq 0}}(q_1(4p_1+p_2)+q_2(-p_1+2p_2)) = \min\{4p_1+p_2, -p_1+2p_2\}$. Stąd, chcąc mieć optymalną strategię dla W musimy rozwiązać następujące zadanie:

$$\begin{array}{ll} \max & \min\{4p_1+p_2, -p_1+2p_2\} \\ & p_1+p_2 = 1 \\ & p_i \geq 0 \qquad\qquad \text{dla } i = 1,2. \end{array} \tag{34}$$

Łatwo widać, że rozwiązanie spełnia równanie $4p_1+p_2 = -p_1+2p_2$, a więc $p_1 = \frac{1}{6}$, $p_2 = \frac{5}{6}$. Taka strategia mieszana gwarantuje graczowi W wygraną o wartości $\frac{3}{2}$.

A jaka jest optymalna strategia (q_1, q_2) dla gracza K? Z analogicznego rozumowania jak dla W dostajemy

$$\begin{array}{ll} \min & \max\{4q_1-q_2, q_1+2q_2\} \\ & q_1+q_2 = 1 \\ & q_i \geq 0 \qquad\qquad \text{dla } i = 1,2. \end{array} \tag{35}$$

Znów widzimy, że rozwiązanie spełnia równanie $4q_1 - q_2 = q_1 + 2p_2$, a więc $q_1 = q_2 = \frac{1}{2}$. Taka strategia mieszana gwarantuje graczowi K „wygraną" o wartości $-\frac{3}{2}$. Czyli tyle samo co dla gracza W! Oznacza to, że jeśli W i K obiorą znalezione przez nas strategie, żadnemu z nich nie będzie opłacało się zmienić jego strategii. (Taką sytuację nazywamy equilibrium Nasha.) Czy otrzymany wynik jest przypadkowy? Oczywiście nie, co potwierdza słynne twierdzenie minimaksowe von Neumanna

Twierdzenie 10. *W każdej 2-osobowej grze o sumie zerowej, istnieją strategie dla graczy W i K oraz liczba $V \in \mathbb{R}$ taka, że strategia gracza W gwarantuje mu wygraną V niezależnie od strategii K oraz strategia K gwarantuje mu wygraną $-V$ niezależnie od strategii W. Innymi słowy, dla dowolnej macierzy A o wymiarach $n \times m$,*

$$\max_{\substack{\sum p_j=1\\ p_j\geq 0}} \min_{\substack{\sum q_i=1\\ q_i\geq 0}} \sum_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} p_j q_i a_{ji} = \min_{\substack{\sum q_i=1\\ q_i\geq 0}} \max_{\substack{\sum p_j=1\\ p_j\geq 0}} \sum_{\substack{1\leq i\leq m\\ 1\leq j\leq n}} p_j q_i a_{ji}. \tag{36}$$

Pokażemy teraz jak powyższe twierdzenie wynika z twierdzenia o dualności PL. Na początek, podobnie jak już zauważyliśmy analizując strategie graczy, zauważmy że dla dowolnego wektora liczb $(\alpha_1, \ldots, \alpha_n)$ mamy

$$\min_{\substack{\sum q_i=1 \\ q_i \geq 0}} \sum_{1 \leq i \leq n} q_i \alpha_i = \min_{1 \leq i \leq n} \alpha_i$$

Podobnie,

$$\max_{\substack{\sum q_i=1 \\ q_i \geq 0}} \sum_{1 \leq i \leq n} q_i \alpha_i = \max_{1 \leq i \leq n} \alpha_i$$

Stąd, (36) jest równoważne poniższej równości.

$$\max_{\substack{\sum p_j=1 \\ p_j \geq 0}} \min_{i=1,\ldots,m} \sum_{j=1}^{n} p_j a_{ji} = \min_{\substack{\sum q_i=1 \\ q_i \geq 0}} \max_{j=1,\ldots,n} \sum_{i=1}^{m} q_i a_{ji} \tag{37}$$

Aby udowodnić prawdziwość (37), rozważmy następujące zadanie o n zmiennych odpowiadające lewej stronie równości (37) (poszukiwaniu optymalnej strategii mieszanej dla gracza W):

$$\begin{array}{ll} \max & \min_{i=1,\ldots,m} \sum_{j=1}^{n} p_j a_{ji} \\ & \sum_{j=1}^{n} p_j = 1 \\ & p_j \geq 0 \qquad\qquad \text{dla } j = 1, \ldots, n. \end{array} \tag{38}$$

To nie jest zadanie programowania liniowego, gdyż funkcja celu nie jest liniowa. W tym wypadku możemy je jednak wyrazić jako następujący równoważny program liniowy:

$$\begin{array}{ll} \max & t \\ & \sum_{j=1}^{n} p_j a_{ji} \geq t \quad \text{dla } i = 1, \ldots, m \\ & \sum_{j=1}^{n} p_j = 1 \\ & p_j \geq 0 \qquad\qquad \text{dla } j = 1, \ldots, n. \end{array} \tag{39}$$

Taki program możemy rozwiązać algorytmem np. simplex (lub algorytmem wielomianowym jeśli mamy taki kaprys), co samo w sobie jest ważną informacją: mamy efektywny sposób znajdowania optymalnych strategii. Jeśli teraz napiszemy program dualny do tego programu (zgodnie z zasadami podanymi w rozdziale 5.4) to dostaniemy następujący program:

$$\begin{array}{ll} \min & s \\ & \sum_{i=1}^{m} q_i a_{ji} \leq s \quad \text{dla } j = 1, \ldots, n \\ & \sum_{i=1}^{m} q_i = 1 \\ & q_i \geq 0 \qquad\qquad \text{dla } i = 1, \ldots, m. \end{array} \tag{40}$$

Powyższy program jest równoważny programowi z prawej strony równości (37). Stąd, silne twierdzenie o dualności implikuje żądaną równość.

5.6.4 Twierdzenie Yao

Rozważmy abstrakcyjny problem algorytmiczny. Niech $\mathcal{A}$ oznacza zbiór wszystkich algorytmów deterministycznych dla danych rozmiaru n oraz niech $\mathcal{I}$ oznacza zbiór wszystkich instancji problemu rozmiaru n. Wówczas $\mathcal{I}$ jest skończony. Zbiór $\mathcal{A}$ nie jest skończony, ale możemy rozważać jedynie algorytmy których pesymistyczny czas nie przekracza jakiejś określonej (dostatecznie dużej) wartości i tak ograniczony zbiór jest już skończony. Niech T będzie macierzą o kolumnach indeksowanych elementami $\mathcal{A}$ i wierszach indeksowanych elementami $\mathcal{I}$, w której element $t_{A,I} = T(A, I)$ ma wartość równą liczbie kroków, które wykonuje algorytm A na instancji I. Ponadto, dla rozkładu prawdopodobieństwa α na algorytmach ze zbioru $\mathcal{A}$ niech zmienna losowa A_α oznacza algorytm wylosowany zgodnie z rozkładem α. Podobnie, dla rozkładu prawdopodobieństwa β na instancjach ze zbioru $\mathcal{I}$ niech zmienna losowa I_β oznacza instancję wylosowaną zgodnie z rozkładem β.

Zastosujmy twierdzenie von Neumanna (a konkretnie równość (37)) do tej macierzy. Otrzymujemy

$$\min_\alpha \max_{I\in\mathcal{I}} \sum_{A\in\mathcal{A}} Pr[A_\alpha = A]t_{A,I} = \max_\beta \min_{A\in\mathcal{A}} \sum_{I\in\mathcal{I}} Pr[I_\beta = I]t_{A,I}.$$

Równoważnie,

$$\min_\alpha \max_{I\in\mathcal{I}} \mathbb{E}[T(A_\alpha, I)] = \max_\beta \min_{A\in\mathcal{A}} \mathbb{E}[T(A, I_\beta)]. \tag{41}$$

Teraz zastanówmy się, co tak naprawdę dostaliśmy. Jak możemy interpretować zmienną losową A_α dla danego rozkładu α? Każdy randomizowany algorytm Las-Vegas (czyli taki, który zawsze zwraca poprawną odpowiedź, ale wykorzystuje losowość i czas jego działania jest zmienną losową) dla naszego problemu możemy traktować jako pewien rozkład prawdopodobieństwa na algorytmach ze zbioru $\mathcal{A}$. Istotnie, każdy taki algorytm korzysta z pewnej liczby bitów losowych. Możemy wartości tych bitów wylosować na samym początku algorytmu, a wówczas jego dalsze działanie jest już zdeterminowane przez te bity (a więc jest zgodne z pewnym algorytmem ze zbioru $\mathcal{A}$).

Skoro A_α oznacza pewien algorytm randomizowany to lewa strona naszej równości oznacza *czas najlepszego (w sensie oczekiwanej złożoności na najgorszych danych) algorytmu randomizowanego* Las-Vegas dla naszego problemu.

Natomiast z prawej strony wybieramy taki rozkład na instancjach, który zmaksymalizuje oczekiwany czas najlepszego algorytmu *deterministycznego*.

Wynika stąd, że równanie (41) daje nam sposób na dowodzenie *dolnych granic* na złożoność algorytmów randomizowanych. Wystarczy mianowicie znaleźć jakiś zestaw „trudnych" instancji i podać taki rozkład prawdopodobieństwa na tym zestawie, że żaden algorytm deterministyczny nie poradzi sobie z nim zbyt dobrze. Ten wniosek znany jest jako twierdzenie Yao.

Twierdzenie 11 (Twierdzenie Yao)**.** *Niech A_α będzie pewnym algorytmem Las-Vegas. Niech β będzie pewnym rozkładem prawdopodobieństwa na instancjach problemu. Wówczas,*

$$\max_{I\in\mathcal{I}} \mathbb{E}[T(A_\alpha, I)] \geq \min_{A\in\mathcal{A}} \mathbb{E}[T(A, I_\beta)]$$

Przykładowe zastosowanie. Rozważmy następujący problem. Mając daną macierzy $n \times n$ o elementach ze zbioru $\{0,1\}$ stwierdzić, czy istnieje kolumna złożona z samych zer. Umówmy się, że interesują nas operacje postaci „sprawdź daną komórkę macierzy". Nasz zestaw „trudnych" instancji to zbiór wszystkich macierzy o elementach ze zbioru $\{0,1\}$, w których każda kolumna zawiera dokładnie jedną jedynkę. W rozkładzie β każda z tych instancji ma równe prawdopodobieństwo, a pozostałe instancje mają prawdopodobieństwo 0. Rozważmy dowolny algorytm deterministyczny A uruchomiony na instancji I_β. Ile czasu potrzebuje A, żeby upewnić się, że j-ta kolumna zawiera 1? Oznaczmy tę zmienną losową przez X. Załóżmy teraz, że elementy postałych komórek (poza j-tą komórką) mają ustalone wartości. Oznaczmy przez A zdarzenie, że tak się stało (mamy $2^{n(n-1)}$ takich zdarzeń). Zbadajmy $E[X|A]$. Gdy pozostałe komórki mają ustalone wartości to A czyta elementy kolumny j-tej w pewnej ustalonej kolejności. Prawdopodobieństwo, że 1-ka będzie akurat w i-tym przeczytanym przez niego elemencie wynosi dokładnie $\frac{1}{n}$. Stąd, warunkowa wartość oczekiwana liczby operacji zużytych na tę kolumnę wynosi $E[X|A] \geq \sum_{i=1}^{n} \frac{1}{n} \cdot i = \frac{n-1}{2}$, niezależnie od tego, jakich wylosowanych wartości dotyczy A. Stąd $EX \geq \frac{n-1}{2}$. Z liniowości wartości oczekiwanej możemy przesumować tę wartość po wszystkich n kolumnach dostając razem, że $\mathbb{E}[T(A, I_\beta)] \geq \frac{n(n-1)}{2}$. Z twierdzenia Yao wnioskujemy, że dowolny algorytm randomizowany Las-Vegas rozwiązujący ten problem dla pewnej instancji wykona co najmniej $\frac{n(n-1)}{2}$ operacji (w sensie wartości oczekiwanej).

5.6.5 przepływy i przekroje

Rozważmy problem maksymalnego przepływu (niedouczony czytelnik jest proszony o przypomnienie sobie definicji problemu maksymalnego przepływu i problemu minimalnego przekroju). Zamiast maksymalizować całkowity przepływ wypływający z s możemy dodać nową krawędź (t,s) o nieskończonej przepustowości oraz zażądać warunku zachowania przepływu (tzn. $\sum_w x_{vw} = 0$) także dla źródła s i ujścia t. Otrzymujemy następujący program:

$$\begin{array}{lll} \text{zmaksymalizuj} & x_{ts} & \\ \text{z zachowaniem warunków} & \sum_{w \in V}(x_{vw} - x_{wv}) = 0 & \text{dla każdego } v \in V \\ & x_{vw} \leq c(v,w) & \text{dla wszystkich } v, w \in V \\ & x_{vw} \geq 0 & \text{dla wszystkich } v, w \in V \end{array} \tag{42}$$

Zauważmy, że warunki zachowania przepływu $\sum_{w \in V}(x_{vw} - x_{wv}) = 0$ możemy zamienić na $\sum_{w \in V}(x_{vw} - x_{wv}) \leq 0$. Istotnie, gdyby w rozwiązaniu dopuszczalnym dla pewnego wierzchołka $\sum_{w \in V}(x_{vw} - x_{wv}) < 0$, czyli mniej wypływa z v niż wpływa do v, to ponieważ całkowity przepływ wpływający do wszystkich wierzchołków jest równy wypływającemu więc z jakiegoś innego wierzchołka q musiałoby więcej wypływać niż wpływać, czyli rozwiązanie nie byłoby dopuszczalne. W ten sposób otrzymaliśmy PL w postaci standardowej:

$$\begin{array}{lll} \text{zmaksymalizuj} & x_{ts} & \\ \text{z zachowaniem warunków} & \sum_{w \in V}(x_{vw} - x_{wv}) \leq 0 & \text{dla każdego } v \in V \\ & x_{vw} \leq c(v,w) & \text{dla wszystkich } v, w \in V \\ & x_{vw} \geq 0 & \text{dla wszystkich } v, w \in V \end{array} \tag{43}$$

Skonstruujmy program dualny. Dla każdego warunku zachowania przepływu program dualny będzie zawierał zmienną z_v. Podobnie, każdemu warunkowi przepustowości w (43) odpowiada zmienna y_{vw} w programie dualnym. Każda zmienna x_{vw} programu (43) występuje raz w warunku zachowania przepływu dla v (ze współczynnikiem 1), raz w warunku zachowania przepływu dla w (ze współczynnikiem -1) oraz raz w warunku przepustowości dla vw (ze współczynnikiem 1). Zmienna x_{vw} będzie więc odpowiadać nierówności o lewej stronie $z_v - z_w + y_{vw}$. Zmienna x_{vw} jest nieujemna więc nierówność będzie postaci $z_v - z_w + y_{vw} \geq ?$, gdzie ? to współczynnik w funkcji celu przy x_{vw}, czyli 1 gdy $vw = ts$ i 0 w przeciwnym przypadku. Możemy już napisać cały program dualny:

$$\begin{array}{lll} \text{zminimalizuj} & \sum_{vw} c(v,w) y_{vw} & \\ \text{z zachowaniem warunków} & z_v - z_w + y_{vw} \geq 0 & \text{dla każdego } vw \neq ts \\ & z_t - z_s + y_{ts} \geq 1 & \\ & y_{vw} \geq 0 & \text{dla wszystkich } v, w \in V \\ & z_v \geq 0 & \text{dla każdego } v \in V \end{array} \tag{44}$$

Zauważmy, że gdyby w optymalnym rozwiązaniu (44) $y_{ts} > 0$ to skoro $c(t,s) = +\infty$, więc (44) jest nieograniczony. Ale wtedy (43), jako program dualny do (44), jest sprzeczny, a przecież zerowy przepływ jest zawsze dopuszczalny. Stąd $y_{ts} = 0$. Uwzględniając tę równość po niewielkich zabiegach kosmetycznych dostajemy:

$$\begin{array}{lll} \text{zminimalizuj} & \sum_{vw} c(v,w) y_{vw} & \\ \text{z zachowaniem warunków} & y_{vw} \geq z_w - z_v & \text{dla każdego } vw \neq ts \\ & z_t - z_s \geq 1 & \\ & y_{vw} \geq 0 & \text{dla wszystkich } v, w \in V \\ & z_v \geq 0 & \text{dla każdego } v \in V \end{array} \tag{45}$$

Zauważmy, że gdyby dołożyć warunki $z_v, y_{vw} \in \{0,1\}$ to dostalibyśmy dokładnie problem minimalnego przekroju! W szczególności, dla minimalnego przekroju $(S, \bar{S})$, rozwiązanie postaci $z_v = 0$ dla $v \in S$, $z_w = 1$ dla $w \in \bar{S}$, $y_{vw} = z_v - z_w$ jest rozwiązaniem dopuszczalnym (45). Na podstawie słabej dualności dostajemy znajomy fakt: max-przepływ $\leq$ min-przekrój.

Gdyby teraz udowodnić, że rozwiązanie optymalne (45) ograniczonego do wartości $\{0,1\}$ jest też rozwiązaniem optymalnym (45), to z twierdzenia o dualności mielibyśmy jako wniosek twierdzenie o maksymalnym przepływie i minimalnym przekroju, tzn. że wartość maksymalnego przepływu jest równa przepustowości minimalnego przekroju. Można tak uczynić, my jednak w inny sposób pokażemy, że twierdzenie o maksymalnym przepływie i minimalnym przekroju jest szczególnym przypadkiem teorii dualności PL.

Wystarczy pokazać, że min-przekrój $\leq$ max-przepływ. Nawet mniej: wystarczy skonstruować dowolny przekrój o przepustowości $\leq$ max-przepływ. Niech $\mathbf{x}^*$ będzie optymalnym rozwiązaniem (43) natomiast $\mathbf{z}^*, \mathbf{y}^*$ optymalnym rozwiązaniem (45). Bez straty ogólności możemy założyć, że $z_s^* = 0$ (jeśli tak nie jest ustawiamy $z_v^* := z_v^* - z_s^*$, otrzymując rozwiązanie dopuszczalne o tej samej wartości funkcji celu). Niech $S = \{v \mid z_v^* < 1\}$ i $T = V - S$. Zauważmy, że $s \in S$ oraz $t \in T$, a więc (S,T) jest przekrojem. Pokażemy, że jego przepustowość jest równa

x^*_{ts}, czyli że $c(S,T) = \sum_{v\in S, w\in T} c(v,w) = x^*_{ts}$. W tym celu rozważmy dwa rodzaje krawędzi (v,w) przecinających przekrój:

1. $v \in S, w \in T$. Wtedy $y^*_{vw} \geq z^*_w - z^*_v > 0$. Stąd $y^*_{vw} \neq 0$ i na podstawie komplementarnych warunków swobody, odpowiedni warunek w programie prymalnym musi być spełniony z równością, czyli $x^*_{vw} = c(v,w)$.

2. $v \in T, w \in S, (v,w) \neq (t,s)$. Wtedy $z^*_v - z^*_w + y^*vw \geq z^*_v - z^*_w > 0$. Stąd warunek $z^*_v - z^*_w + y^*vw \geq 0$ w programie (45) nie jest spełniony z równością, więc na mocy komplementarnych warunków swobody odpowiadająca mu zmienna x^*_{vw} w (43) jest równa 0.

Pokazaliśmy, że

$$c(S,T) = \sum_{\substack{v\in S\\ w\in T}} x^*_{vw} - \sum_{\substack{w\in T, v\in S\\ (w,v)\neq(t,s)}} x^*_{wv}.$$

Jeszcze kilka przekształceń[9]

$$\begin{aligned}
c(S,T) &= \sum_{\substack{v\in S\\ w\in T}} x^*_{vw} - \sum_{\substack{w\in T, v\in S\\ (w,v)\neq(t,s)}} x^*_{wv} \\
&= x^*_{ts} + \sum_{\substack{v\in S\\ w\in T}} x^*_{vw} - \sum_{\substack{w\in T\\ v\in S}} x^*_{wv} \\
&= x^*_{ts} + \left(\sum_{\substack{v\in S\\ w\in T}} x^*_{vw} + \sum_{v,w\in S} x^*_{vw}\right) - \left(\sum_{\substack{w\in T\\ v\in S}} x^*_{wv} + \sum_{w,v\in S} x^*_{wv}\right) \\
&= x^*_{ts} + \sum_{\substack{v\in S\\ w\in V}} x^*_{vw} - \sum_{\substack{w\in V\\ v\in S}} x^*_{wv} \\
&= x^*_{ts} + \sum_{v\in S} \underbrace{\sum_{w\in V} (x^*_{vw} - x^*_{vw})}_{0} = x^*_{ts}.
\end{aligned}$$

6 Podziękowania

Szczególnie gorąco dziękuję Adamowi Iwanickiemu za uważną lekturę wstępnej wersji tego tekstu i liczne uwagi oraz Marcinowi Musze za pomocne dyskusje. Dziękuję także studentom wykładu z algorytmiki: Jakubowi Bartodziejowi, Mateuszowi Machalicy i Marcinowi Ziombskiemu za wychycenie literówek i drobnych usterek.

[9] właściwie już widać, że z prawej strony mamy przepływ płynący przez przekrój, a to jest równe po prostu x^*_{ts}, ale kto o tym pamięta?

Literatura

- Notatki do wykładu „Advanced Algorithms" E. Demiane'a i D. Kargera (2003) `http://courses.csail.mit.edu/6.854/fall03/`
- Notatki do wykładu „Advanced Algorithms" M. Goemans'a (1994) `http://www-math.mit.edu/~goemans/notes-lp.ps`
- O zastosowaniach PL do algorytmów aproksymacyjnych można poczytać w V. Vazirani „Algorytmy Aproksymacyjne" WNT 2005.
- O programowaniu liniowym (z perspektywy algorytmu Simplex) można też poczytać w Cormen, Leiserson, Rivest, Stein „Wprowadzenie do algorytmów", Wydanie ≥ 6, WNT 2004.