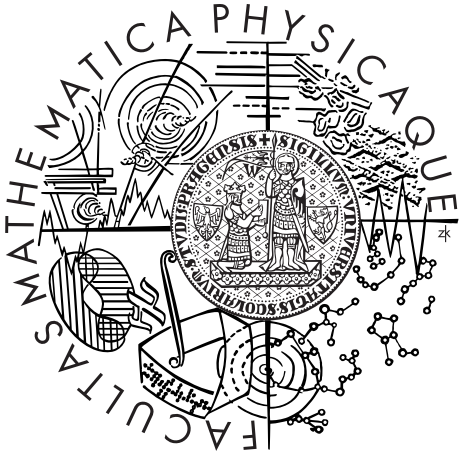
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Viktor Popovič

Optimalizace toku grafem

Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce:	doc. RNDr. Petr Lachout, CSc.
Studijní program:	Matematika
Studijní obor:	Finanční matematika

Praha 2013

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Univerzita Karlova v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V ..... dne ..... Podpis autora

Název práce: Optimalizace toku grafem

Autor: Viktor Popovič

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedoucí bakalářské práce: doc. RNDr. Petr Lachout, CSc., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: Optimalizace je důležitá každodenní činnost, ať už chceme maximalizovat efektivitu nebo minimalizovat náklady. Mnoho problémů z praxe umíme převést do teorie grafů a následně optimalizovat. V této práci se budeme věnovat dopravnímu problému, který spočívá v uspokojení požadavků všech odběratelů za co nejnižší cenu. Další je problém maximálního toku, kde chceme sítí, v které má každá hrana kapacitní omezení, přepravit co nejvíce komodity (ropa, plyn, ...). Také se podíváme na jeho alternaci v případě, že spolu s maximalizací toku chceme zároveň minimalizovat náklady. Na řešení těchto problémů si zavedeme numerické algoritmy jako metodu řádkových a sloupcových čísel, značkovací algoritmus, algoritmus nejkratší zvětšující se cesty a Preflow-Push algoritmus. Jejich funkčnost si nakonec předvedeme na příkladě, kde se potvrdí správnost algoritmů a jejich rozdíly.

Klíčová slova: graf; optimální tok; numerický algoritmus

Title: Optimization of flow in graph

Author: Viktor Popovič

Department: Department of Probability and Mathematical Statistics

Supervisor: doc. RNDr. Petr Lachout, CSc., Department of Probability and Mathematical Statistics

Abstract: When it comes to maximization of effectively or minimizing of cost, optimization represents the key activity. There is a number of practical examples that can be implemented into Theory of Graphs and subsequently optimized. This thesis includes the introduction to transportation problem where the consumer demand is met by the lowest price. Also there is maximum flow problem which is to transfer maximum of commodity (petroleum, gas...) through the network where each edge has a capacity restriction. We will also look into the alternative situations where we will maximize the flow along with minimizing of cost. To resolve these problems we will establish numeric algorithms like distribute method, labeling algorithm, shortest augmented path algorithm, and Preflow-Push algorithms. We will also illustrate functionality on example which confirms appropriate application of algorithms and differences among them.

Keywords: graph; optimal flow; numerical algorithm

Názov práce: Optimalizácia toku grafom

Autor: Viktor Popovič

Katedra: Katedra pravděpodobnosti a matematické statistiky

Vedúci bakalárskej práce: doc. RNDr. Petr Lachout, CSc., Katedra pravděpodobnosti a matematické statistiky

Abstrakt: Optimalizácia je dôležitá každodenná činnosť, či už chceme maximalizovať efektivitu alebo minimalizovať náklady. Veľa problémov z praxe vieme previesť do teórie grafov a následne optimalizovať. V tejto práci sa budeme venovať dopravnému problému, ktorý spočíva v uspokojení požiadaviek všetkých odberateľov za čo najnižšiu cenu. Ďalší je problém maximálneho toku, kde chceme sieťou, v ktorej má každá hrana kapacitné obmedzenie, prepraviť čo najviac komodity (ropa, plyn, ...). Takisto sa pozrieme na jeho alternáciu v prípade, že spolu s maximalizáciou toku chceme zároveň minimalizovať náklady. Na riešenie týchto problémov si zavedieme numerické algoritmy ako metódu riadkových a stĺpcových čísiel, značkovací algoritmus, algoritmus najkratšej zväčšujúcej cesty a Preflow-Push algoritmus. Ich funkčnosť si nakoniec predvedieme na príklade, kde sa potvrdí správnosť algoritmov a ich rozdiely.

Kľúčové slová: graf; optimálny tok; numerický algoritmus

Obsah

Úvod

V tejto bakalárskej práci sa budeme venovať optimalizácii toku grafom.

Na úvod si pripomenieme niektoré kľúčové pojmy zo súvisiacich matematických disciplín ako sú diskrétna matematika a optimalizácia.

Jedným z dôležitých problémov je dopravný problém, ktorému venujeme druhú kapitolu. Problém spočíva v minimalizácií dopravných nákladov pri preprave tovaru od dodávateľa do skladov odberateľov. Musíme však dodržať podmienky, ktoré sú obmedzené ponukou dodávateľa a musia sa naplniť požiadavky skladov. Predstavíme si matematickú formuláciu tohto problému, t.j. prechod na úlohu lineárneho programovania. Ukážeme si tri algoritmy na hľadanie prípustného bázického riešenia a spomenieme ich výhody a nevýhody. Potom si predstavíme metódu riadkových a stĺpcových čísel, ktorá postupnými iteráciami nájde optimálne riešenie z prípustne bázického riešenia. Kapitolu ukončíme praktickou ukážkou metódy severozápadného rohu, metódy minimálnej ceny a nakoniec metódy riadkových a stĺpcových čísel na príklade.

V tretej kapitole si zadefinujeme pojmy ako tok a sieť a takisto aj úlohu o maximálnom toku. Tá sa dá v praxi predstaviť ako sieť ropovodov alebo plynovodov, kde časti potrubia majú isté kapacitné obmedzenia a my hľadáme maximálne množstvo komodity, ktorú môžeme nimi prepraviť. Ukážeme si Ford a Fulkersonov algoritmus, ktorý sa nazýva algoritmom zväčšujúcich ciest a značkovací algoritmus. Patria medzi najjednoduchšie algoritmy, ktoré riešia úlohu maximálneho toku, no ich nevýhodou je vysoká časová náročnosť. Ďalej si ukážeme algoritmus najkratšej zväčšujúcej cesty, ktorý je o niečo komplikovanejší, ale jeho časová náročnosť je výrazne nižšia. Ako posledný algoritmus v tretej kapitole uvedieme Preflow-Push algoritmus, v ktorom si budeme musieť pozmeniť definíciu toku a opäť dostaneme algoritmus s nízkou časovou náročnosťou. Nakoniec si uvedieme rôzne modifikácie úlohy o maximálnom toku a spôsob akým ich previesť na pôvodnú úlohu, ktorú už vieme riešiť.

V štvrtej kapitole budeme opäť maximalizovať tok, no zároveň budeme chcieť minimalizovať cenu. V tejto časti si uvedieme algoritmus záporných cyklov, ktorý nadviaže na algoritmy predstavené v tretej kapitole.

V poslednej kapitole použijeme značkovací algoritmus, algoritmus najkratšej zväčšujúcej cesty a Preflow-Push algoritmus na príklade inšpirovanom praxou, kde uvidíme rozdielny prístup algoritmov k riešeniu úlohy.

Kapitola 1

Teoretické základy

1.1 Disktrétna matematika

Aby sme pochopili vety, definície, dôkazy a algoritmy, ktoré budeme rozoberať v ďalších kapitolách, budeme si musieť pripomenúť niektoré časti diskrétnej matematiky. Všetky pojmy a vety, o ktoré sa budeme opierať a ktoré budeme používať, sa nachádzajú v tejto sekcii a ďalej v texte budeme predpokladať ich znalosť za samozrejmú.

Definícia 1 (graf, orientovaný graf)**. Graf** *G pozostáva z neprázdnej konečnej množiny V a množiny E, ktorá je množinou dvojbodových podmnožín množiny V. Prvok množiny V nazývame vrchol (vertex) a prvok množiny E nazývame hrana (edge). Zvyčajné značenie je $G = (V, E)$.*
Ak E je množinou usporiadaných dvojbodových podmnožín množiny V, potom $G = (V, E)$ nazývame **orientovaný graf**.

Definícia 2 (bipartitný graf)**.** *Graf $G = (V, E)$ sa nazýva* **bipartitný**, *pokiaľ vieme množinu V rozdeliť na dve podmnožiny V', V'' také, že každá hrana z množiny E spája vrchol z množiny V' s vrcholom z množiny V''.*

Definícia 3 (podgraf)**.** *Graf G' je* **podgrafom** *grafu G, ak platí, že $V(G') \subseteq V(G)$ a $E(G') \subseteq E(G)$.*

Definícia 4 (cyklus)**. Cyklom** *v grafe $G = (V, E)$ rozumieme postupnosť*

$$(v_0, e_1, v_1, e_2, \ldots, e_{t-1}, v_{t-1}, e_t, v_0),$$

kde $v_i \in V(G)$ pre $\forall i = 1, ..., t-1$, navzájom rôzne a $e_i = v_{i-1}v_i \in E(G)$.

Poznámka. Iné pomenovanie cyklu v grafe je kružnica.

Definícia 5 (súvislosť)**.** *Graf G nazývame* **súvislý**, *pokiaľ pre každé dva vrcholy x a y existuje cesta z x do y.*

S týmito znalosťami o grafoch si môžme pripomenúť základné poznatky o stromoch.

Definícia 6 (strom)**.** *Súvislý graf, ktorý neobsahuje cyklus, nazývame* **strom**.

Lemma 1 (o koncovom vrchole)**.** *Každý strom s aspoň dvoma vrcholmi obsahuje najmenej dva vrcholy, z ktorých vychádza práve jedna hrana (vrchol stupňa 1). Takýto vrchol nazývame koncovým.*

Nasledujúce dve vety sú prevzaté z [8] strany 141, 142.

Veta 2 (postupná výstavba stromu)**.** *Pre daný graf G a jeho koncový vrchol v sú nasledujúce tvrdenia ekvivalentné:*

i) G *je strom,*

ii) $G - v$ *je strom.*

Dôkaz. V [8] strana 141.

❑

Veta 3 (charakterizácia stromu)**.** *Pre graf $G = (V, E)$ sú nasledujúce podmienky ekvivalentné:*

i) G *je strom.*

ii) (jednoznačnosť cesty)
Pre každé dva vrcholy $x, y \in V$ existuje práve jedna cesta z x do y.

iii) (minimálna súvislosť)
Graf G je súvislý, vynechaním ľubovoľnej hrany vznikne nesúvislý graf.

iv) (maximálny graf bez kružníc)
Graf G neobsahuje kružnicu, každý graf, ktorý vznikne z G pridaním hrany (t.j. graf tvaru $G + e$, kde $e \in \binom{V}{2} \setminus E$), už bude kružnicu obsahovať.

v) (Eulerov vzorec)
G je súvislý a $|V| = |E| + 1$.

Dôkaz. Jednotlivé kroky dôkazu sú podrobne popísané v [8] strany 142, 143.

❑

Definícia 7 (kostra)**.** *Nech $G = (V, E)$ je súvislý graf.* **Kostrou** *potom rozumieme graf $K = (V, E')$, pre ktorý platí:*

i) $E' \subseteq E$,

ii) T *je stromom.*

Takýto teoretický úvod z oblasti diskrétnej matematiky nám zatiaľ bude stačit.

1.2 Optimalizácia

Okrem pojmov z diskrétnej matematiky si pripomenieme niektoré pojmy z optimalizácie, ktorých znalosť v dalších kapitolách budeme opäť považovať za samozrejmú.

Definícia 8. Úloha lineárneho programovania v štandardnom tvare *je zadaná:*

$$min\{\boldsymbol{c}^\top \boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b},\, \boldsymbol{x} \geq \boldsymbol{0}\}, \tag{1.1}$$

kde $\boldsymbol{b} \in \mathbb{R}^p$, $\boldsymbol{A} \in \mathbb{R}^{p\times q}$ *sú podmienky, pri ktorých minimalizujeme a* $\boldsymbol{c} \in \mathbb{R}^q$ *reprezentuje výraz, ktorý minimalizujeme.*

Nasledujúca definícia je prevzatá z [5] strana 22.

Definícia 9. *Uvažujme maticu* $\boldsymbol{A} \in \mathbb{R}^{p\times q}$. *Vyberme nejakú množinu indexov premenných* $L \subset \{1,\, 2,\, \ldots,\, q\}$ *tak, aby obsahovala práve* m *rôznych položiek. Vezmeme príslušné stĺpce matice* $\boldsymbol{A}$ *a zostavíme z nich štvorcovú maticu* $\boldsymbol{A}_L = (A_{j,i},\, j \in \{1,\, 2,\, \ldots,\, p\},\, i \in L)$.

- *Pokiaľ je matica* $\boldsymbol{A}_L$ *regulárna, potom hovoríme, že L je* **báza úlohy** (1.1).
- *Ak L je bázou úlohy 1.1, potom k nej patrí* $\boldsymbol{x}(L) \in \mathbb{R}^n$, *ktoré je jednoznačne určené podmienkami* $\boldsymbol{A}\boldsymbol{x}(L) = \boldsymbol{b}$ *a* $\boldsymbol{x}(L)_i = 0$ *pre všetky* $i \notin L$. *Bod* $\boldsymbol{x}(L)$ *nazývame* **bázické riešenie** *príslušné k báze L.*
- *Nech L je bázou úlohy* (1.1) *a bázické riešenie* $\boldsymbol{x}(L)$ *je nezáporné. Potom sa jedná o* **prípustnú bázu**.
- *Nech L je bázou úlohy* (1.1) *a príslušné bázické riešenie* $\boldsymbol{x}(L)$ *je optimálnym riešením úlohy* (1.1). *Potom sa jedná o* **optimálnu bázu**.
- *Hovoríme, že* $\boldsymbol{x}$ *je* **bázické riešenie** *úlohy* (1.1), *pokiaľ existuje báza L taká, že* $\boldsymbol{x} = \boldsymbol{x}(L)$.
- *Hovoríme, že* $\boldsymbol{x}$ *je* **prípustné bázické riešenie** *úlohy* (1.1), *pokiaľ existuje prípustná báza L taká, že* $\boldsymbol{x} = \boldsymbol{x}(L)$.
- *Hovoríme, že* $\boldsymbol{x}$ *je* **optimálne bázické riešenie** *úlohy* (1.1), *pokiaľ existuje optimálna báza L taká, že* $\boldsymbol{x} = \boldsymbol{x}(L)$.

Nasledujúcu definíciu sme prevzali z [5] strany 31, 32.

Definícia 10 (duálna úloha)**.** *Nech je daná matica* $\boldsymbol{A}$ *dimenzie* $(p \times q)$, *p-rozmerný vektor* $\boldsymbol{b}$, *q-rozmerný vektor* $\boldsymbol{c}$ *a disjunktné rozklady množín indexov* $I_\geq \cup I_\leq \cup I_\in = \{1,\, 2,\, \ldots,\, q\}$, $J_\geq \cup J_\leq \cup J_= = \{1,\, 2,\, \ldots,\, p\}$. *Potom dvojica úloh*

lineárneho programovania

$$
\begin{aligned}
&\textit{minimalizovať} && \sum_{i=1}^{q} c_i x_i && \\
&\textit{za podmienok} && \sum_{i=1}^{q} A_{j,i} x_i \geq b_j && \textit{pre } \forall j \in J_{\geq}, \\
& && \sum_{i=1}^{q} A_{j,i} x_i \leq b_j && \textit{pre } \forall j \in J_{\leq}, \\
& && \sum_{i=1}^{q} A_{j,i} x_i = b_j && \textit{pre } \forall j \in J_{=}, \\
& && x_i \geq 0 && \textit{pre } \forall i \in I_{\geq}, \\
& && x_i \leq 0 && \textit{pre } \forall i \in I_{\leq}, \\
& && x_i \in \mathbb{R} && \textit{pre } \forall i \in I_{\in},
\end{aligned}
$$

$$
\begin{aligned}
&\textit{maximalizovať} && \sum_{j=1}^{p} b_j x_j && \\
&\textit{za podmienok} && \sum_{j=1}^{p} A_{j,i} y_j \leq c_i && \textit{pre } \forall i \in I_{\geq}, \\
& && \sum_{j=1}^{p} A_{j,i} y_j \geq c_i && \textit{pre } \forall i \in I_{\leq}, \\
& && \sum_{j=1}^{p} A_{j,i} y_j = c_i && \textit{pre } \forall i \in I_{\in}, \\
& && y_j \geq 0 && \textit{pre } \forall j \in J_{\geq}, \\
& && y_j \leq 0 && \textit{pre } \forall j \in J_{\leq}, \\
& && y_j \in \mathbb{R} && \textit{pre } \forall j \in J_{=},
\end{aligned}
$$

sa nazýva **dvojicou duálnych úloh** *lineárneho programovania.*

Kapitola 2

Dopravná úloha

Najprv sa budeme venovať dopravnej úlohe, ktorá je špeciálnou úlohou lineárneho programovania. Jedná sa o tok bipartným grafom, ktorého hrany majú špeciálnu orientáciu.

2.1 Zadefinovanie dopravnej úlohy

Zaveďme si slovnú a matematickú formuláciu dopravnej úlohy.

Máme p skladov $s_1, \ldots, s_p$, v ktorých sa nachádza rovnorodý výrobok v množstvách $a_1, \ldots, a_p$, ktorý je treba rozviezť odberateľom $t_1, \ldots, t_q$, ktorí požadujú množstvá $b_1, \ldots, b_q$. Náklady na prepravu zo skladu s_i k odberateľovi b_j sú $c_{i,j}$. Cieľom je minimalizovať celkové dopravné náklady a naplniť požiadavky odberateľov. Nech $x_{i,j}$ je hľadané množstvo tovaru prevážané zo skladu s_i k odberateľovi t_j, potom matematická formulácia má nasledujúci tvar.

Nech $a_i,\ b_j \geq 0,\ c_{i,j} \in \mathbb{R}$, potom

$$\begin{aligned}
&\text{minimalizovať} && \sum_{i=1}^{p}\sum_{j=1}^{q} c_{i,j}x_{i,j}, && && (2.1)\\
&\text{za podmienok} && \sum_{j=1}^{q} x_{i,j} = a_i && (i = 1,\ \ldots,\ p),\\
& && \sum_{i=1}^{p} x_{i,j} = b_j && (j = 1,\ \ldots,\ q),\\
& && x_{i,j} \geq 0 && (i = 1,\ \ldots,\ p;\ j = 1,\ \ldots,\ q).
\end{aligned}$$

Vyššie sformulovaná dopravná úloha sa nazýva vyvážená kvôli rovnostiam $\sum_{j=1}^{q} x_{i,j} = a_i$, pre $\forall i = 1, \ldots, p$ a $\sum_{i=1}^{p} x_{i,j} = b_j$ pre $\forall j = 1, \ldots, q$, teda dopyt odberateľov sa rovná ponuke skladov. Pri hľadaní prípustného a optimálneho riešenia môžeme používať tento tvar vďaka nasledujúcej lemme.

Lemma 4. *Nasledujúce podmienky sú pre úlohu* (2.1) *ekvivalentné:*
i) platí $\sum_{i=1}^{p} a_i = \sum_{j=1}^{q} b_j$,

ii) má prípustné riešenie,

iii) má optimálne riešenie.

Dôkaz. V [4] strany 189, 190. ❑

Poznámka. Je zrejmé, že v prípade $\sum_{i=1}^{p} a_i < \sum_{j=1}^{q} b_j$ neexistuje prípustné riešenie.

Na záver tejto časti si napíšeme dopravnú úlohu (2.1) ako úlohu lineárneho programovania v štandardnom tvare:

$$min\{\boldsymbol{c}^\top \boldsymbol{x} : \tilde{\boldsymbol{A}}\boldsymbol{x} = \tilde{\boldsymbol{b}}, \boldsymbol{x} \geq \boldsymbol{0}\}. \tag{2.2}$$

2.2 Hľadanie prípustného bázického riešenia

Sústava rovníc $\tilde{\boldsymbol{A}}\boldsymbol{x} = \tilde{\boldsymbol{b}}$ (výraz (2.2)) má špecifický tvar. Pre lepšiu predstavu sa pozrieme, ako vyzerá. Použijeme rozmery $p = 3$, $q = 4$.

$$\boldsymbol{x} = (x_{1,1},\, x_{1,2},\, x_{1,3},\, x_{1,4},\, x_{2,1},\, x_{2,2},\, x_{2,3},\, x_{2,4},\, x_{3,1},\, x_{3,2},\, x_{3,3},\, x_{3,4})^\top,$$

$$\tilde{\boldsymbol{A}} = \left(\begin{array}{cccc|cccc|cccc} 1&1&1&1&0&0&0&0&0&0&0&0\\ 0&0&0&0&1&1&1&1&0&0&0&0\\ 0&0&0&0&0&0&0&0&1&1&1&1\\ \hline 1&0&0&0&1&0&0&0&1&0&0&0\\ 0&1&0&0&0&1&0&0&0&1&0&0\\ 0&0&1&0&0&0&1&0&0&0&1&0\\ 0&0&0&1&0&0&0&1&0&0&0&1 \end{array}\right), \tilde{\boldsymbol{b}} = \left(\begin{array}{c} a_1\\ a_2\\ a_3\\ \hline b_1\\ b_2\\ b_3\\ b_4 \end{array}\right).$$

Matica $\tilde{\boldsymbol{A}} \in \mathbb{R}^{p+q \times pq}$ má v každom stĺpci práve 2 jednotky a vidíme systém ich rozmiestnenia. Hodnosť matice $\tilde{\boldsymbol{A}}$ je $p + q - 1$, viď. [4] strana 190, čo využijeme v ďalšej úvahe.
Označme si stĺpce matice $\tilde{\boldsymbol{A}}$ ako $\tilde{\boldsymbol{a}}_{\boldsymbol{11}}$, $\tilde{\boldsymbol{a}}_{\boldsymbol{12}}$, …, $\tilde{\boldsymbol{a}}_{\boldsymbol{1q}}$, $\tilde{\boldsymbol{a}}_{\boldsymbol{21}}$, …, $\tilde{\boldsymbol{a}}_{\boldsymbol{pq}}$. $\tilde{\boldsymbol{A}}$ môžeme priradiť grafu $K_{p,q}$ a opačne, kde hrane $s_i t_j$ prináleží stĺpec $\tilde{\boldsymbol{a}}_{\boldsymbol{ij}}$ pre $\forall i = 1, \ldots, p$ a $\forall j = 1, \ldots, q$. Pomocou nasledujúcej lemmy transformujeme problém hľadania prípustnej bázy dopravnej úlohy na problém hľadania kostry grafu $K_{p,q}$.

Lemma 5. *Sústava S pozostávajúca z $p + q - 1$ stĺpcov matice $\tilde{\boldsymbol{A}}$ je lineárne nezávislá práve vtedy, keď podgraf grafu $K_{p,q}$ s hranami prináležiacimi sústave S je stromom.*

Dôkaz. V [4] strana 191. ❑

Stačí teda nájsť kostru grafu $K_{p,q}$ a určiť počet prepravovaného tovaru $x_{ij} \geq 0$ na jej hranách. Samozrejme nesmieme pri konštrukcii kostry zabudnúť na podmienky dané a_i a b_j. Z lemmy o koncovom vrchole vieme, že v strome existujú aspoň dva koncové body, teda body, z ktorých vedie práve jedna hrana. To znamená, že ak majú byť splnené podmienky dopravnej úlohy, tak na tejto hrane platí, napríklad $x_{ij} = a_i$ (alebo $x_{ij} = b_j$, záleží na tom, či je koncovým vrcholom sklad alebo odberateľ). Z vety o postupnej výstavbe stromu vieme, že po odstránení koncového vrcholu nám zostane strom. Odstráňme teda vrchol a_i a príslušnú hranu (už poznáme množstvo tovaru prepravené touto hranou). Dostali sme o jeden vrchol menšiu dopravnú úlohu. Na tejto myšlienke sú vybudované nasledujúce algoritmy.

Algoritmus 1 (všeobecný)**.** *Priradíme* $i \in \{1, \ldots, p\} := \boldsymbol{I}$, $j \in \{1, \ldots, q\} := \boldsymbol{J}$.

1. krok *Ľubovoľne vyberieme* $i \in \boldsymbol{I}$, $j \in \boldsymbol{J}$.

2. krok *Priradíme* $x_{ij} := min\{a_i, b_j\}$ *a* $a_i := a_i - x_{ij}$, $b_j := b_j - x_{ij}$.

3. krok *Ak platí, že* $a_i = 0$ *alebo* $b_j = 0$, *tak* $\boldsymbol{I} := \boldsymbol{I} \setminus \{i\}$ *alebo* $\boldsymbol{J} := \boldsymbol{J} \setminus \{j\}$. *V prípade, že* $a_i = 0$ *a zároveň* $b_j = 0$, *odstránime index z ľubovoľnej množiny s aspoň dvoma prvkami, ak neexistuje taká množina, odstránime z ľubovoľnej.*

4. krok *Vrátime sa na* **1. krok**. *Algoritmus skončíme pri vyprázdnení jednej z množín.*

Existujú viaceré modifikácie tohoto algoritmu, niektoré si teraz uvedieme.

Algoritmus 2 (metóda severozápadného rohu)**.**

1. krok *Vyberieme* $i \in \boldsymbol{I}$, $j \in \boldsymbol{J}$, *aby boli čo najmenšie.*

2. krok *Kroky 2, 3, 4 sú rovnaké ako v Algoritme 1.*

Výhodou metódy severozápadného rohu je jednoduchosť a rýchlosť výpočtov. To si názorne ukážeme na riešení Príkladu 1, taktiež tam demonštrujeme tabuľkový zápis, pomocou ktorého počítame.

Uvedieme si ďalší alogritmus.

Algoritmus 3 (metóda minimálnej ceny)**.**

1. krok *Vyberieme* $i \in \boldsymbol{I}$, $j \in \boldsymbol{J}$ *tak, aby bolo* $c_{i,j}$, *teda cena za prepravu, čo najmenšia možná.*

2. krok *Kroky 2, 3, 4 sú rovnaké ako v Algoritme 1.*

Poznámka. Tento algoritmus sa vyskytuje aj s menom indexová metóda.

Keďže vyberáme najnižšiu cenu, môžeme očakávať riešenie bližšie ku optimálnemu, no vo všeobecnosti to neplatí.

Algoritmus 4 (Vogelova aproximačná metóda)**.**

1. krok *Pre každý stĺpec a každý riadok tabuľky vypočítame rozdiel medzi najmenšou a druhou najmenšou cenou. Zvolíme riadok alebo stĺpec, kde je táto hodnota najväčšia a v ňom $c_{i,j}$, $i \in \boldsymbol{I}$, $j \in \boldsymbol{J}$, ktoré je najmenšie.*

2. krok *Kroky 2, 3, 4 sú rovnaké ako v Algoritme 1.*

Tento algoritmus dáva v praxi často riešenia blízke optimálnemu, no opäť to teoreticky nie je zaručené. Nevýhodou je množstvo výpočtov v porovnaní s Algoritmami 2 a 3.

Veta 6. *Algoritmus 1 zaručí bázické prípustné riešenie dopravnej úlohy po $p+q-1$ iteráciach.*

Dôkaz. Vďaka Lemme 5 nám stačí dokázať, že hrany $s_i t_j$, zodpovedajúce vybraným dvojiciam indexov (i,j), tvoria kostru grafu $K_{p,q}$ a že zložky x_{ij} tvoria prípustné riešenie. Vetu dokážeme pomocou matematickej indukcie.
Nech $p = q = 1$, potom na hrane $s_1 t_1$ jednoducho určíme $x_{ij} = a_i = b_j$, teda je zrejmé, že to platí.
Indukčným predpokladom je, že tvrdenie platí pre $p + q - 1 > 2$.
Dokážeme tvrdenie pre $p + q > 2$. Ak postupujeme podľa algoritmu, teda po vybraní prvého indexu odstránime koncový vrchol stromu, dostaneme dopravnú úlohu s $p + q - 1$ vrcholmi, čo je náš indukčný predpoklad.

❑

2.3 Testovanie optimality

Pri testovaní optimality použijeme modifikáciu simplexovej metódy pre vyvážený dopravný problém. Táto modifikácia sa nazýva metóda riadkových a stĺpcových čísiel.
Najprv sa pozrime, ako vyzerá duálna úloha k úlohe (2.1).

$$\begin{aligned} max \quad & \sum_{i=1}^{p} a_i u_i + \sum_{j=1}^{q} b_j v_j \\ \text{za podmienok} \quad & u_i + v_j \leq c_{i,j} \qquad \forall\, i = 1, \ldots, p \text{ a } j = 1, \ldots, q. \end{aligned}$$

Označme si $J = \{(i,j) : i = 1, \ldots, p;\ j = 1, \ldots, q\}$. Uvedieme algoritmus. Tento algoritmus je prevzatý z [5] strana 43.

Algoritmus 5 (metóda riadkových a stĺpcových čísel)**.**

1. krok *Nájdeme počiatočnú primárne prípustnú bázu $L_0 \subset J$, $card(L_0) = p + q - 1$ pomocou jedného z algoritmov uvedených v predchádzajúcej sekcii.*

2. krok *Máme k dispozícii primárne prípustnú bázu* $L_k \subset J$, $card(L_k) = p+q-1$. *Vyriešime sústavu rovníc*

$$u_i + v_j = c_{i,j} \qquad \forall (i,j) \in L_k.$$

Výsledkom sú riadkové čísla ${u_i}^*$, $i = 1, \ldots, p$ *a stĺpcové čísla* ${v_j}^*$, $j = 1, \ldots, q$. *Riešenie nie je dané jednoznačne, ale súčty* ${u_i}^* + {v_j}^*$ *sú jednoznačné.*

3. krok *Pokiaľ*

$${u_i}^* + {v_j}^* \leq c_{i,j} \qquad \forall (i,j) \in L_k,$$

tak prejdeme na **5. krok**.
V opačnom prípade nájdeme $(\iota,\kappa) \in J$ *také, že* ${u_\iota}^* + {v_\kappa}^* > c_{\iota,\kappa}$ *a prejdeme na* **4. krok**.

4. krok *Nájdeme cestu* (i_0,j_0), (i_1,j_1), …, $(i_{2D},j_{2D}) \in J$ *takú, že*

- $i_0 = i_{2D} = \iota$, $j_0 = j_{2D} = \kappa$.
- *Pre* $d = 0, 1, \ldots, D-1$ *je* $i_{2d+1} = i_{2d+2}$ *a* $j_{2d} = i_{2d+1}$.
- (i_1,j_1), (i_2,j_2), …, $(i_{2D-1},j_{2D-1}) \in L_k$.

Nájdeme $\Delta \in \{0, 1, \ldots, D-1\}$ *tak, že*

$$x(L_k)_{i_{2\Delta+1},j_{2\Delta+1}} = min\{x(L_k)_{i_{2d+1},j_{2d+1}} : d = 0,1,\ldots,D-1\}.$$

Položíme $L_{k+1} = (L_k \cup \{(\iota,\kappa)\} \setminus \{i_{2\Delta+1},j_{2\Delta+1}\})$. *Potom* L_{k+1} *je primárne prípustná báza a vraciame sa na* **2. krok**.

5. krok *Nájdená báza* L_k *je optimálna a preto* $x(L_k)$ *je optimálnym riešením dopravnej úlohy* (2.1).

Príklad 1. Použili sme zadanie z [5] strana 43. Nech $p = 3$ a $q = 4$ (máme tri sklady s_1, s_2, s_3 a štyroch odberateľov t_1, t_2, t_3 a t_4). Ceny $c_{i,j}$ za prepravu uvedieme v matici $\boldsymbol{c}$. Podmienky a_i a b_j zadáme vo vektoroch $\boldsymbol{a}$ a $\boldsymbol{b}$.

$$\boldsymbol{a} = \begin{pmatrix} 5 \\ 4 \\ 3 \end{pmatrix}, \boldsymbol{b} = \begin{pmatrix} 4 \\ 2 \\ 4 \\ 2 \end{pmatrix}, \boldsymbol{c} = \begin{pmatrix} 2 & 2 & 1 & 1 \\ 1 & 1 & 2 & 1 \\ 1 & 2 & 1 & 2 \end{pmatrix}.$$

Prvý krok Algoritmu 5 hovorí, že máme nájsť primárne prípustnú bázu. Aby sme si názorne ukázali rozdiely medzi algoritmami, použijeme dva z nich. Ako prvý Algoritmus 2 (metóda severozápadného rohu) a po ňom Algoritmus 3 (metóda minimálnej ceny). Pre obidva uvedieme tabuľku už s konečnou bázou. Postupnosť jednotlivých krokov môžeme vidieť pri postupnej zmene podmienok (čo je druhý krok v obidvoch algoritmoch) a naviac uvedieme poradie, v ktorom sme získavali prvky bázy.

	t_1	t_2	t_3	t_4	$\boldsymbol{a}$
s_1	4	1			~~5,1~~
s_2		1	3		~~4,3~~
s_3			1	2	~~3,2~~
$\boldsymbol{b}$	~~4~~	~~2,1~~	~~4,1~~	~~2~~	

S Algoritmom 2 sme dostali takúto postupnosť prvkov v báze: (1,1), (1,2), (2,2), (2,3), (3,3), (3,4). Môžeme vidieť, že tento algoritmus je naozaj jednoduchý a prehľadný, v tabuľke začneme v pravom hornom rohu a postupne prechádzame k ľavému dolnému.
Hodnota účelovej funkcie je:

$$4 \times 2 + 1 \times 2 + 1 \times 1 + 3 \times 2 + 1 \times 1 + 2 \times 2 = 22.$$

	t_1	t_2	t_3	t_4	$\boldsymbol{a}$
s_1			4	1	~~5,1~~
s_2	4				~~4~~
s_3	0	2		1	~~3,1~~
$\boldsymbol{b}$	~~4~~	~~2~~	~~4~~	~~2,1~~	

S Algoritmom 3 sme dostali takúto postupnosť prvkov v báze: (2,1), (3,1), (1,3), (1,4), (3,2), (3,4). Keďže matica $\boldsymbol{c}$ obsahuje iba jednotky a dvojky, mohli sme dostať aj iné poradie alebo inú bázu, pretože pri vyberaní najmenšej ceny vyberáme z viacerých možností. Všimnime si takisto prítomnosť 0 v báze. Tá vznikla tak, že v prvej iterácii sa nám vynulovali obidve podmienky, teda mohli sme si vybrať, z ktorej množiny odstránime index a keďže $c_{1,3} = 1$, čo je najnižšia cena, tak sme ju následne vybrali do bázy.
Hodnota účelovej funkcie je:

$$4 \times 1 + 0 \times 1 + 4 \times 1 + 1 \times 1 + 2 \times 2 + 1 \times 2 = 15.$$

Vidíme, že obidva algoritmy sa skončili po $p + q - 1 = 6$ iteráciach. V tomto prípade je k optimálnemu riešeniu bližšie algoritmus 3 a bázu, ktorú sme s ním našli, budeme používať ďalej.

Dopočítame si riadkové a stĺpcové čísla, ktoré zapíšeme do prehľadnej tabuľky.

	v	0	1	1	1	
u		t_1	t_2	t_3	t_4	$\boldsymbol{a}$
0	s_1	2 0	2 1	1 1 4	1 1 1	5
1	s_2	1 1 4	1 !2	2 2	1 !2	4
1	s_3	1 1 0	2 2 2	1 !2	2 2 1	3
	$\boldsymbol{b}$	4	2	4	2	

V ľavom hornom rohu máme $c_{i,j}$, v pravom hornom rohu sa nachádza súčet ${u_i}^* + {v_j}^*$. Teraz vidíme, že kritérium optimality bolo porušené pre 3 dvojice indexov (výkričník). Nech $(\iota,\kappa) = (2,2)$. Nájdeme cestu: (2,2) → (3,2) → (3,1) → (2,1) → (2,2). Zistíme, že (2,2) bude v novej báze namiesto (3,2). Prepočítame tabuľku a vypočítame nové riadkové a stĺpcové čísla.

	v	0	0	1	1	
u		t_1	t_2	t_3	t_4	$\boldsymbol{a}$
		2 0	2 0	1 1	1 1	
0	s_1			4	1	5
		1 1	1 1	2 2	1 !2	
1	s_2	2	2			4
		1 1	2 1	1 !2	2 2	
1	s_3	2			1	3
	$\boldsymbol{b}$	4	2	4	2	

Vidíme, že kritérium optimality bolo porušené 2 dvoch prípadoch (výkričník). Nech $(\iota,\kappa) = (2,4)$. Nájdeme cestu: (2,4) → (3,4) → (3,1) → (2,1) → (2,4). Zistíme, že (2,4) bude v novej báze namiesto (3,4). Prepočítame tabuľku.

	v	0	0	0	0	
u		t_1	t_2	t_3	t_4	$\boldsymbol{a}$
		2 1	2 1	1 1	1 1	
1	s_1			4	1	5
		1 1	1 1	2 1	1 1	
1	s_2	1	2		1	4
		1 1	2 1	1 1	2 1	
1	s_3	3			1	3
	$\boldsymbol{b}$	4	2	4	2	

Vidíme, že teraz sú splnené všetky kritéria optimality, čo znamená, že sme našli optimálne riešenie.
Hodnota účelovej funkcie:

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 4 \times 1 + 1 \times 1 + 1 \times 1 = 12.$$

Kapitola 3

Úloha o maximálnom toku

V tejto kapitole sa budeme venovať maximalizácii toku v sieti a predstavíme si základné algoritmy.

3.1 Úvod

Ako prvé si zadefinujeme pojem siete.

Definícia 11 (sieť)**.** *Orientovaný graf $G = (V, E)$ s vyznačeným vrcholom zdroja s a vrcholom výstupu t a s kapacitnou funkciou u: $E(G) \to \mathbb{N}_0$ sa nazýva* **sieť**.

Funkcia $u(E)$ priradzuje každej orientovanej hrane kapacitu (v praxi to môže byť šírka potrubia ...), ktorá v konečnom dôsledku predstavuje maximálne množstvo komodity, ktoré môže cez túto orientovanú hranu prejsť za jednu časovú jednotku. Často budeme používať značenie $u(e_{ij}) = u_{ij}$, $ij \in E(G)$. Označme si $U = max\{u_{ij}: ij \in E\}$.

Majme sieť $G = (V, E)$. V tejto kapitole predpokladáme, že pre každú hranu $ij \in E$ platí, že hrana $ji \in E$. Nestratíme tým na obecnosti, pretože Definícia 11 povoľuje nulové kapacity hrán.

V predchádzajúcej kapitole sme pracovali s pojmom tok intuitívne, ako presun výrobku zo skladu k odberateľovi. V tejto kapitole budeme potrebovať presnejší prístup a preto si tento pojem zadefinujeme (tok môže prezentovať rôzne komodity z praxe ako ropa, plyn, piesok, ... alebo aj elektrický prúd).

Definícia 12 (tok)**.** *Funkciu x: $E(G) \to \mathbb{N}_0$ s vlastnosťami:*

$$0 \leq x_{ij} \leq u_{ij} \qquad \forall ij \in E(G), \tag{3.1}$$

$$\sum_{\{j:ij \in E\}} x_{ij} - \sum_{\{j:ji \in E\}} x_{ji} = 0 \qquad \forall i \in V(G) \setminus \{s,t\} \tag{3.2}$$

nazývame **tok**.

Prvá podmienka zaručuje dodržanie kapacít orientovaných hrán ij a druhá podmienka nám zaručuje to, že množstvo komodity, ktoré do vrcholu pritečie, vrchol aj opustí. Vidíme, že množstvo komodity počas toku medzi vstupom s a výstupom t nič nemôže zmeniť. Zadefinujeme si pojem veľkosť toku.

Definícia 13 (veľkosť toku)**.** *Číslo*

$$f(x) = \sum_{\{j:sj \in E\}} x_{sj} - \sum_{\{j:js \in E\}} x_{js}$$

nazývame **veľkosťou toku**.

Úloha o maximálnom toku znamená **maximalizovať f(x)** (takýto tok sa nazýva maximálny).

Posledný veľmi dôležitý pojem, ktorý si zavedieme v tejto časti kapitoly je reziduálna sieť.

Definícia 14. *Nech x je tok v sieti. Pre každú orientovanú hranu v sieti zavedieme pojem* **reziduálna kapacita** r_{ij}*, ktorá predstavuje maximálnu veľkosť doplnkového toku (tok, ktorý môžeme pridať k aktuálnemu toku z vrcholu i do vrcholu j využívac hrany ij a ji). Platí, že* $r_{ij} = u_{ij} - x_{ij} + x_{ji}$*. Reziduálna kapacita je zložená z dvoch prvkov, prvý je nevyužitá kapacita toku na orientovanej hrane a druhý je zrušenie prípadného toku z vrcholu j do vrcholu i.*
Sieť zostavenú z orientovaných hrán ohodnotených kladnými reziduálnymi kapacitami (z pôvodného toku x) nazývame **reziduálna sieť**.

3.2 Algoritmus zväčšujúcich ciest a značkovací algoritmus

Algioritmus zväčšujúcich ciest predstavili Ford a Fulkerson. Technicky to nie je algoritmus, keďže nezaviedli, ako máme hľadať zväčšujúce cesty, pozrime sa, v čom spočíva.

Algoritmus 6.

1. krok *Zavedieme tok* $x = 0$ *a vytvoríme reziduálnu sieť.*

2. krok *Nájdeme orientovanú cestu P zo zdroja s do výstupu t. Ak žiadna neexistuje, ideme na* **4. krok**.

3. krok *Priradíme* $\Delta = min\{r_{ij} : ij \in P\}$*, zvýšime tok x pozdĺž P o* Δ *a prepočítame ohodnotienie hrán v našej reziduálnej sieti. Vrátime sa na* **2. krok**.

4. krok *Získali sme maximálny tok f(x).*

Poznámka. Nulový tok, ktorý sme si zaviedli v 1. kroku, je prípustný tok.

Definícia 15 (zväčšujúca cesta)**.** *Orientovaná cesta začínajúca v s a končiaca v t v reziduálnej sieti sa nazýva* **zväčšujúca cesta**.

V značkovacom algoritme si už ukážeme konkrétne kroky ako hľadať zväčšujúce cesty a ako zistíme, že už žiadna neexistuje. Vrcholy budeme označkovávať číslom vrcholu, z ktorého sme do nich prišli, teda pre vrchol j, do ktorého sme sa dostali z vrcholu i bude značka vyzerať: $pred(j) = i$.

Algoritmus 7.

0. krok *Zavedieme tok* $x = 0$.

1. krok *Vytvoríme reziduálnu sieť. Označkujeme si vrchol s (zdroj).*

2. krok *Vyberieme označkovaný vrchol i. Pre všetky hrany ij vychádzajúce z vrcholu i (nezabúdajme, že pracujeme v reziduálnej sieti) skontrolujeme, či je vrchol j označkovaný. Ak nie je označkovaný a* $r_{ij} > 0$, *potom* $pred(j) = i$.

3. krok *Ak t nie je označkované a neexistuje žiadna hrana spájajúca označkované a neoznačkované vrcholy, ideme na* **5. krok**. *Ak je vrchol t (výstup) označkovaný, postúpime na* **4. krok**, *inak sa vrátime na* **2. krok**.

4. krok *Využijeme značky* pred *pri každom vrchole, aby sme našli zväčšujúcu cestu P z s do t. Zväčšíme tok x po ceste P o* $\Delta = min\{r_{ij} : ij \in P\}$, *zmažeme všetky označenia* pred *a vrátime sa na* **1. krok**.

5. krok *Našli sme maximálny tok, ktorý si z reziduálnych kapacít vieme vypočítať ako*

$$\begin{aligned} x_{ij} &= u_{ij} - r_{ij}, \qquad & x_{ji} &= 0 \qquad & pre\ u_{ij} > r_{ij} \\ x_{ji} &= r_{ij} - u_{ij}, \qquad & x_{ij} &= 0 \qquad & pre\ u_{ij} \leq r_{ij} \end{aligned}$$

Nevýhodou tohoto algoritmu je jeho vysoká časová náročnosť. Každý značkovací proces v jeho priebehu preskúma každý vrchol najviac raz. Keďže značkovací alogoritmus zväčší veľkosť toku o najmenej jedničku, horná hranica počtu iterácií je nU v sieti s n vrcholmi. Pri veľkom U môžeme dosahovať veľmi veľké počty iterácií. Druhou nevýhodou tohto algoritmu je fakt, že vždy zmažeme celé označkovanie vrcholov. Môžeme pri tom mazať niektoré informácie, ktoré by nám mohli pomocť pri dalších iteráciach. Vidíme, že je tu priestor na zlepšovanie, čo si ukážeme v ďalších algoritmoch.

V závere tejto časti ešte uvedieme vetu, ktorá dokazuje korektnosť vyššie uvedených algoritmov.
Nech $S \subset V(G)$, $\bar{S} = V(G) \setminus S$, $s \in S$, $t \in \bar{S}$. Označme $(S,\bar{S})_G$ množinu orientovaných hrán, ktoré spájajú každé $i \in S$ s $j \in \bar{S}$. Túto množinu nazývame rezom. Kapacitu rezu označujeme $c(S,\bar{S})_G$ a definujeme ju ako súčet individuálnych kapacít všetkých hrán rezu, teda $f(S,\bar{S})_G = \sum_{i \in S, j \in \bar{S}} u_{ij}$. Rez s minimálnou kapacitou sa nazýva minimový.

Veta 7 (o maximálnom toku a minimovom reze)**.** *V ľubovoľnej sieti sa maximálny tok rovná kapacite minimového rezu.*

Dôkaz. V [1] strana 74.

❑

3.3 Algoritmus najkratšej zväčšujúcej cesty

Ako názov tohto algoritmu naznačuje, budeme si musieť zaviesť pojem vzdialenosti, aby sme vedeli nájsť najkratšiu zväčšujúcu cestu. V celej tejto časti budeme pracovať s reziduálnou sieťou.

Definícia 16. *Zavedieme* **vzdialenostnú funkciu** $d : V \to \mathbb{N}_0$, *pre ktorú sú splnené nasledujúce podmienky*

$$d(t) = 0,$$
$$d(i) \leq d(j) + 1 \qquad \forall ij \in E \text{ také, že: } r_{ij} > 0.$$

Pod $d(i)$ rozumieme označkovanie vzdialenosti vrcholu i.

Platí, že $d(i) \leq k$ pre ľubovoľnú cestu dĺžky k v reziduálnej sieti začínajúcej vo vrchole i. To nám dáva spodné ohraničenie dĺžky najkratšej zväčšujúcej cesty.

Definícia 17. *Orientovanú hranu ij v reziduálnej sieti nazveme* **prijateľnou**, *pokiaľ platí $d(i) = d(j) + 1$. Ostatné orientované hrany sú neprijateľné.*

Algoritmus, ktorý uvedieme, bude zvyšovať tok po prijateľných cestách, ktoré sú zložené z prijateľných hrán a spájajú zdroj s výstupom. Pre ľubovoľnú prijateľnú cestu dĺžky k bude platiť $d(s) = k$. Spodnou hranicou dĺžky každej cesty zo zdroja s k výstupu t je $d(s)$ a alogritmus bude zvyšovať tok po najkratších cestách. Dôležité je poznamenať, že v sieti s n vrcholmi sa dĺžka najdlhšej možnej cesty z s do t rovná $n - 1$. Tento poznatok využijeme pri určení, kedy ukončiť algoritmus.

Algoritmus 8.

1. krok *Zavedieme tok $x = 0$ a vytvoríme reziduálnu sieť. Prevedieme spätné prehľadávanie do šírky začínajúc vo výstupe t a stanovíme $d(i)$, $\forall i \in V$. Ako aktuálny vrchol i^* označíme vrchol s.*

2. krok *Ak je $d(s) \geq n$, algoritmus sa skončil a našli sme maximálny tok. Inak skontrolujeme, či existujú prijateľné hrany vychádzajúce z vrcholu i^*. Ak áno, ideme na* **4. krok**, *ak nie, ideme na* **5. krok**.

3. krok *Skontrolujeme, či $i^* = t$, ak áno, ideme na* **6. krok**, *ak nie, vrátime sa na* **2. krok**.

4. krok *Nech prijateľná hrana spája vrcholy i^* a j^*. Označkujeme $pred(j^*) = i^*$, zmeníme aktuálny vrchol $i^* = j^*$ a vrátime sa na* **3. krok**.

5. krok *Zmeníme vzdialenosť $d(i^*) = min\{d(j) + 1 : ij \in E(i^*) \text{ a } r_{ij} > 0\}$. Ak $i^* \neq s$, tak zmeníme aktuálny vrchol $i^* = pred(i^*)$ a vrátime sa na* **3. krok**.

6. krok *Pomocou označkovania pred nájdeme zväčšujúcu cestu P zo zdroja s do výstupu t a zväčšíme po nej tok o $\Delta = min\{r_{ij} : ij \in P\}$. Zmeníme aktuálny vrchol na $i^* = s$ a vrátime sa na* **2. krok**.

Štvrtý krok v tomto algoritme predstavuje posun na ďalší vrchol v hľadanej najkratšej prijateľnej ceste. Naopak v piatom kroku sa o jeden vrchol vrátime, keď neexistuje už žiadna prijateľná orientovaná hrana a následne sa zvýši hodnota vzdialenosti $d(i^*)$ vrcholu, z ktorého sme sa vrátili tak, aby z neho viedla aspoň jedna prijateľná hrana.

Výhoda tohoto algoritmu je nižšia časová náročnosť a to konkrétne $O(n^2m)$.

3.4 Preflow-Push algoritmus

Definícia 18. *Posielanie toku o veľkosťi Δ po ceste pozostávajúcej z k orientovaných hrán sme v predchádzajúcucich algoritmoch vnímali ako jednu operáciu. Túto operáciu si teraz rozložíme na k základných operácií, ktoré pošlú tok veľkosti Δ po jednotlivých hranách cesty. Túto základnú operáciu budeme volať* **push**.

Keďže jedna push operácia dokáže poslať tok na práve jednu hranu, nastáva spor s (3.2). Tento algoritmus povoľuje, aby tok na vstupe do vrcholu presahoval tok, ktorý z vrcholu vychádza. Každý takýto tok budeme nazývať preflow.

Definícia 19. Preflow *x je funkcia z $x : E \to R$, pre ktorú platí* (3.1) *a*

$$\sum_{\{j:ji\in E\}} x_{ji} - \sum_{\{j:ij\in E\}} x_{ij} \geq 0 \qquad \forall i \in V(G) \setminus \{s,t\}.$$

Pre daný preflow x definujeme prebytok pre každý vrchol $i \in V(G) \setminus \{s,t\}$ ako

$$e(i) = \sum_{\{j:ji\in E\}} x_{ji} - \sum_{\{j:ij\in E\}} x_{ij}$$

Vrchol s kladnou hodnotou prebytku budeme nazývať aktívnym vrcholom.

Dve základné operácie v tomto algoritme budú push operácia na prijateľnej hrane a zmena hodnoty vzdialenosti d(i) pri vrcholoch. Preflow-Push algoritmus využíva iba lokálne informácie. Pri každej iterácii algoritmu (okrem inicializácie a ukončenia) sieť obsahuje aspoň jeden aktívny vrchol a cieľom každej iterácie je poslať jeho prebytok bližšie k výstupu t. Blízkosť je chápaná ako označenie vrcholov d(i), ktoré sme si vysvetlili skôr v tejto kapitole. Podobne ako pri predchádzajúcom algoritme budeme využívať iba prijateľné hrany. Ak nevieme z nejakého vrcholu poslať prebytok ďalej (kvôli neexistencii prijateľných hrán), zvýšime d(i) vrcholu tak, aby sme dostali aspoň jednu novú prijateľnú hranu. Algoritmus sa ukončí, ak sieť neobsahuje žiadne aktívne vrcholy.
Pri používaní tohto algoritmu si pri každom vrchole $i \in V(G) \setminus \{s,t\}$ musíme pamätať okrem vzdialenosti $d(i)$ aj hodnotu prebytku $e(i)$.

Algoritmus 9.

1. krok *Zavedieme tok $x = 0$ a vytvoríme reziduálnu sieť. Prevedieme spätné prehľadávanie do šírky začínajúc vo výstupe t a stanovíme $d(i)$, $\forall i \in V$. Ďalej priradíme $x_{sj} = u_{sj}$ každej orientovanej hrane sj vychádzajúcej z vrcholu s a $d(s) = n$.*

2. krok *Ak neexistuje žiaden aktívny vrchol, algoritmus sa skončil a našli sme maximálny tok.*

3. krok *Zoberieme aktívny vrchol i. Ak existuje prijateľná hrana ij, potom prevedieme push operáciu veľkosti $\delta = min\{e(i), r_{ij}\}$ na hrane ij. V opačnom prípade zmeníme $d(i) = min\{d(j) + 1 : ij \in E(i)$ a $r_{ij} > 0\}$ a vrátime sa na* **2. krok**.

Poznámka. Nesmieme zabudnúť, že po každej push operácii musíme prepočítať hodnoty prebytkov e(i) a e(j).

Tento algoritmus má časovú zložitosť $O(n^2m)$, avšak najlepšie algoritmy založené na push-preflow predčia algoritmy, ktoré hľadajú najkratšiu zväčšujúcu cestu.

3.5 Modifikácie

Na základe vyššie vysvetleného modelu siete je možné formulovať modifikované úlohy o maximálnom toku sieťou.

V prípade, že sieť má viacej ako jeden zdroj s, môžeme si ju previesť na úlohu, ktorú sme riešili v tejto kapitole. Spravíme tak pomocou pridania nového zdroja, ktorý spojíme pomocou umelých hrán s nekonečne veľkými kapacitami s množinou zdrojov v pôvodnej úlohe. To isté platí, ak by mala úloha viacero výstupov.

Ďalšou modifikáciou je, ak máme zadané okrem kapacít hrán aj kapacitu vrcholov. V tom prípade uzol i s kapacitou k_i nahradíme dvoma uzlami i' a i'', medzi ktorými vedie hrana s kapacitou k_i. Uzol i' je k sieti pripojený rovnakými hranami, aké vchádzali do uzla i a i'' je obdobne pripojený k sieti s rovnakými uzlami, aké vychádzali z uzla i. Týmto sme previedli úlohu na požadovaný tvar.

Kapitola 4

Úloha o najlacnejšom maximálnom toku

Ďalšou modifikáciou úlohy o maximálnom toku je, že na každej hrane okrem kapacity zavedieme aj cenu $c_{i,j} \in \mathbb{R}$ prepravy za jednotku komodity. Budeme hľadať minimálnu cenu maximálneho toku v sieti.

4.1 Algoritmus záporných cyklov

V tomto algoritme sa opäť budeme opierať o reziduálnu sieť. Z pôvodnej siete vytvoríme reziduálnu sieť nasledujúcim spôsobom. Každú orientovanú hranu $ij \in E$ s cenou $c_{i,j}$ nahradíme dvoma orientovanými hranami ij a ji, pre ktoré platí nasledovné: orientovaná hrana ij má cenu $c_{i,j}$ a reziduálnu kapacitu $r_{ij} = u_{ij} - x_{ij}$ a orientovaná hrana ji má cenu $-c_{i,j}$ a reziduálnu kapacitu $r_{ji} = x_{ij}$. Reziduálna sieť pozostáva jedine z orientovaných hrán, ktoré majú kladné reziduálne kapacity. Problém môže nastať, ak máme v pôvodnom grafe orientované hrany ij aj ji. Potom by v reziduálnej sieti mohla nastať situácia, že máme dve hrany idúce z i do j a dve hrany idúce z j do i s rôznymi cenami. Tento problém vyriešime tým, že zavedieme pomocné vrcholy medzi i a j, podobne ako, keď sme prevádzali modifikovanú úlohu s kapacitami pre vrcholy na pôvodný problém.

Definícia 20. Cena cyklu *(orientovaného)je definovaná ako súčet cien na jeho orientovaných hranách.*

Veta 8. *Tok x veľkosti $f(x) = h$ na sieti má minimálnu cenu (spomedzi tokov s veľkosťou h) práve vtedy, ak v reziduálnej sieti neexistuje cyklus zápornej ceny.*

Dôkaz. V [4] strana 209. ❑

Na tejto vete je založený algoritmus záporných cyklov.

Algoritmus 10.

1.krok *Nájdeme maximálny tok x v sieti pomocou niektorých algoritmov z tretej kapitoly.*

2.krok *V reziduálnej sieti zistíme existenciu cyklu zápornej ceny K. Ak žiaden neexistuje, tok x je hľadaný maximálny tok.*

3.krok *Určíme* $\delta = min\{r_{ij} : ij \in K\}$ *a zmeníme tok v* K *o príslušné* δ. *Vrátime sa na* **2.krok**.

Kapitola 5

Aplikácia algoritmov na príklade

V tejto kapitole si predvedieme funkčnosť troch algoritmov na úlohe o hľadaní maximálneho toku. Príklad je inšpirovaný reálnymi datami zo sieťe plynovodov v Európe. Množstvá plynu, ktoré jednotlivými úsekmi plynovodu pretečú, sú ovplyvnené dohodami medzi štátmi. Veľkosť potrubia je štandardizovaná, teda neurčuje hodnoty kapacít medzi jednotlivými mestami. Pre jednotlivé úseky plynovodu sa nám nepodarilo zistiť aktuálne kapacity. Hodnoty kapacít sme si preto nastavili sami. Finálna sieť, s ktorou budeme pracovať, je uvedená na Obrázku 5.1. Ako zdroj s si označíme mesto Novy Urengoy, kde sa zemný plyn ťaží a chceme ho prepraviť do Berlína, čo bude náš výstup t. Pri ostatných mestách sme uviedli vedľa názvu aj číslo, ktorým na daný vrchol budeme odkazovať v postupe riešenia jednotlivými algoritmami.

5.1 Aplikácia značkovacieho algoritmu

Zavedieme si tok $x = 0$ a vytvoríme reziduálnu sieť. Vrchol s považujeme za označkovaný. Ako prvú preveríme hranu $s2$. Platí, že vrchol 2 nie je označkovaný a $r_{s2} > 0$, teda následne ho označkujem a $pred(2) = s$. Týmto spôsobom postupujeme cez vrchol 6 až do vrcholu t. Keďže sme označkovali výstup, využijeme značky $pred$ a nájdeme zväčšujúcu cestu P z s do t, na ktorej zväčšíme tok x o $min\{r_{ij} : ij \in P\}$, čo je v našom prípade $r_{s2} = 8$. Po tomto kroku zmažeme všetko značkovanie a začneme odznova. Uvedieme zväčšujúce cesty, ktoré sme dostali postupnými iteráciami.
Zväčšujúca cesta (s, 3, 4, 5, 7, t), na ktorej zvýšim tok o $r_{7t} = 6$.
Zväčšujúca cesta (s, 3, 4, 5, 6, t), na ktorej zvýšim tok o $r_{34} = 2$.
Zväčšujúca cesta (s, 4, 5, 6, t), na ktorej zvýšim tok o $r_{45} = 3$.
Zväčšujúca cesta (s, 4, 2, 6, t), na ktorej zvýšim tok o $r_{s4} = 2$.

Dostali sme sa do bodu, keď z označkovaného vrcholu s v reziduálnej sieti, vedie jediná hrana s kladnou reziduálnou kapacitou do vrcholu 3. Ten označkujeme, z neho už neexistuje žiadna hrana, ktorá vedie do neoznačkovaného vrcholu. Algoritmus teda ukončíme a máme maximálny tok o veľkosti 21, $x_{s2} = 8$, $x_{s3} = 8$, $x_{s4} = 5$, $x_{26} = 10$, $x_{25} = 0$, $x_{34} = 8$, $x_{42} = 2$, $x_{45} = 11$, $x_{56} = 5$, $x_{57} = 6$, $x_{6t} = 15$, $x_{7t} = 6$.

5.2 Aplikácia algoritmu najkratšej zväčšujúcej cesty

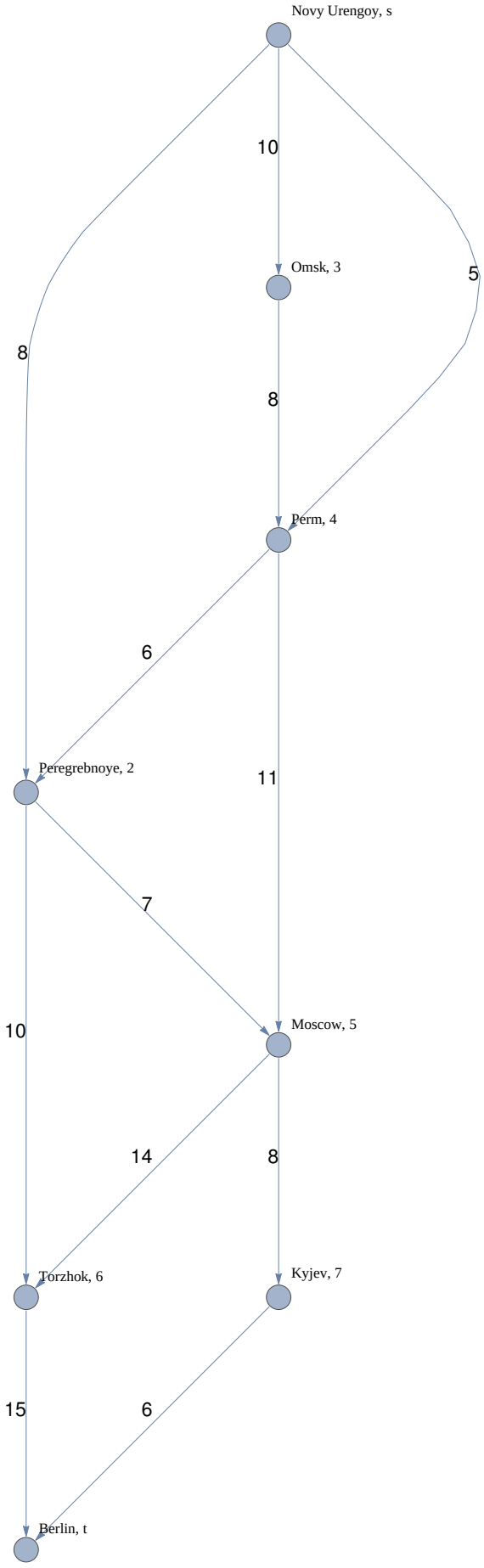
Zavedieme si tok $x = 0$ a vytvoríme reziduálnu sieť. Prevedieme spätné prehľadávanie do šírky začínajúc vo výstupe t a dostávame: $d(s) = 3$, $d(2) = 2$, $d(3) = 4$, $d(4) = 3$, $d(5) = 2$, $d(6) = 1$, $d(7) = 1$, $d(t) = 0$. Ako aktuálny vrchol si označíme s. Pracujeme zo sieťou, ktorá obsahuje 8 vrcholov, teda $n = 8$ a vidíme, že $d(s) = 3 < n = 8$, preto môžeme v algoritme pokračovať. Hrana $s2$ spája vrchol s, $d(s) = 3$ a vrchol 2, $d(2) = 2$, teda je prijateľná. Označíme $pred(2) = s$ a za aktuálny vrchol zvolíme vrchol 2. Pokračujeme v hľadaní prijateľných hrán, prejdeme cez vrchol 1 až do vrcholu t. Pomocou značiek $pred$ nájdeme najkratšiu zväčšujúcu cestu P zo zdroja s do výstupu t. Zväčšíme na nej tok x o $min\{r_{ij} : ij \in P\}$, čo je v našom prípade $r_{s2} = 8$. Označíme vrchol s ako aktuálny a opakujeme algoritmus. Uvedieme aké najkratšie zväčšovacie cesty sme dostali ako aj zmeny vzdialenosti bodov v reziduálnej sieti.
Najkratšia zväčšujúca cesta (s, 4, 5, 7, t), na ktorej zvýšim tok o $r_{s3} = 5$.
Ďalej z bodu s už neexistuje prijateľná hrana a preto $d(s) = min\{d(j) + 1 : ij \in E(i^*)$ a $r_{ij} > 0\} = 4 + 1 = 5$. Pokračujeme v algoritme.
Najkratšia zväčšujúca cesta (s, 3, 4, 5, 7, t), na ktorej zvýšim tok o $r_{7t} = 1$.
Najkratšia zväčšujúca cesta (s, 3, 4, 5, 6, t), na ktorej zvýšim tok o $r_{45} = 5$. Pri tejto iterácií sme vrcholu 7 museli zvýšiť vzdialenosť na $d(7) = 3$.
Najkratšia zväčšujúca cesta (s, 3, 4, 2, 6, t), na ktorej zvýšim tok o $r_{34} = r_{26} = r_{6t} = 2$.

Dostali sme sa do situácie, kde ako aktuálny vrchol máme označený vrchol s, $d(s) = 5$ a jediná prijateľná hrana v reziduálnej sieti vedie do vrcholu 3, $d(3) = 4$, ktorý označíme ako aktuálny vrchol. Z vrcholu 3 žiadna prijateľná hrana nevedie a preto $d(3) = 5 + 1 = 6$ a opäť označím s ako aktuálny vrchol. Tentoraz z vrcholu s nevedie žiadna prijateľná hrana a preto $d(s) = 6 + 1 = 7$. Tento proces sa zopakuje ešte raz, až bude splnené, že $d(s) \geq n$ a algoritmus ukončíme. Máme maximálny tok o veľkosti 21, $x_{s2} = 8$, $x_{s3} = 8$, $x_{s4} = 5$, $x_{26} = 10$, $x_{25} = 0$, $x_{34} = 8$, $x_{42} = 2$, $x_{45} = 11$, $x_{56} = 5$, $x_{57} = 6$, $x_{6t} = 15$, $x_{7t} = 6$.

5.3 Aplikácia Preflow-Push algoritmu

Zavedieme si tok $x = 0$ a vytvoríme reziduálnu sieť. Prevedieme spätné prehľadávanie do šírky začínajúc vo výstupe t a dostávame: $d(s) = 3$, $d(2) = 2$, $d(3) = 4$, $d(4) = 3$, $d(5) = 2$, $d(6) = 1$, $d(7) = 1$, $d(t) = 0$. Nech $x_{s2} = 8$, $x_{s3} = 10$, $x_{s4} = 5$ a $d(s) = n = 8$. Pri vrcholoch 2, 3, 4 dostávame prebytky $e_2 = 8$, $e_3 = 10$, $e_4 = 5$ a teda všetky tri sú aktívnymi vrcholmi. Zoberieme si vrchol 2. Z neho vedie prijateľná hrana do vrcholu 6. Prevedieme push operáciu o veľkosti $min\{e(i), r_{ij}\}$, čo je v našom prípade 8. Prebytok vo vrchole 2 bude $e_2 = 0$ a $e_6 = 8$. Z vrcholu 6 sa stal aktívny, existuje z neho prijateľná cesta do t, a preto z neho následne vykonáme push operáciu o veľkosti 8 a $e_6 = 0$. Pripomeňme si, že pre vrcholy s a t prebytky nie sú definované.
Ako ďalší aktívny vrchol, s ktorým budeme pracovať, si vyberme vrchol 3. Existuje z neho prijateľná cesta do vrcholu 4, vykonáme push operáciu o veľkosti 8 a dostaneme $e_3 = 2$, $e_4 = 13$.
Týmto spôsobom postupujeme ďalej a postupne zvyšujeme tok a posúvame ho

push operáciami bližšie k výstupu t. Vzdialenosť vrcholu 7 sa zmení na $d(7) = 3$, až sa dostaneme do situácie, kde jediným aktívnym vrcholom je vrchol 3, $e_3 = 2$. Nevedú z neho žiadne prijateľné hrany a preto zvýšim jeho vzdialenosť na $d(3) = min\{d(j) + 1 : ij \in E(i) \text{ a } r_{ij} > 0\}$. Jedinou hranou v reziduálnej sieti s kladnou reziduálnou kapacitou je hrana $3s$ a preto $d(3) = 8 + 1 = 9$. Vrchol 3 je stále aktívnym, ale tentoraz už existuje prijateľná hrana. Vykonáme push operáciu o veľkosti 2 z vrcholu 3 do vrcholu s, $e_3 = 0$. Týmto v reziduálnej sieti nezostal ani jeden aktívny vrchol a dostávame maximálny tok o veľkosti 21, $x_{s2} = 8$, $x_{s3} = 8$, $x_{s4} = 5$, $x_{26} = 10$, $x_{25} = 0$, $x_{34} = 8$, $x_{42} = 2$, $x_{45} = 11$, $x_{56} = 5$, $x_{57} = 6$, $x_{6t} = 15$, $x_{7t} = 6$.



Obr. 5.1: Sieť plynovodu

Záver

V prvej časti tejto práce sme sa zamerali na algoritmy používané pri hľadaní prípustne bázického riešenia dopravnej úlohy. V teoretickej časti sme uviedli jednoduchosť metódy severozápadného rohu ako výhodu tohto algoritmu, čo sa v praktickej časti potvrdilo. Pri metóde najnižšej ceny sme ako výhodu spomenuli tendenciu dostávať riešenia, ktoré sú bližšie k optimálnemu riešeniu, čo praktická časť potvrdila (neplatí všeobecne), ale nevýhodou je zložitejší algoritmus. Pri hľadaní optimálneho riešenia sme uviedli iba jeden algoritmus - metódu riadkových a stĺpcových čísel. Praktická časť potvrdila funkčnosť tohto algoritmu. Mohli sme v práci uviesť variácie tejto metódy pri degenerovaných dopravných úlohách alebo ďalšie algoritmy na hľadanie optimálneho riešenia. Považujeme však miesto vyhradené dopravnej úlohe za dostatočné.

V druhej časti sme sa venovali úlohe o maximálnom toku. Teoreticky sme si zaviedli značkovací algoritmus, algoritmus najkratšej zväčšujúcej cesty a Preflow-Push algoritmus. Použili sme ich na príklade inšpirovanom praxou a preukázali sme ich funkčnosť v hľadaní maximálneho toku. Bolo by zaujímavé v práci pokračovať naprogramovaním uvedených algoritmov, čo by umožňovalo efektívne riešiť oveľa zložitejšie príklady, na ktorých by sa lepšie demonštrovali rozdiely v časových náročnostiach jednotlivých algoritmov.

Priestor na ďalšie rozšírenie tejto práce vidíme aj v algoritmoch na riešenie úlohy najlacnejšieho maximálneho toku.

Literatúra

[1] Ravindra K. Ahuja, Thomas L. Magnanti, James B. Orlin. Network Flows. Massachusetts Institute of Technology, Cambridge, 1998. No. 2059-88.

[2] Ian Anderson. A First Course in Discrete Mathematics. Springer, London, 2001. ISBN 1-85233-236-0.

[3] Norman L. Biggs. Discrete Mathematics. Second edition. Oxford University Press, Oxford, 2002. ISBN 0-19-850717-8.

[4] Jitka Dupačová, Ján Plesník, Milan Vlach. Lineárne programovanie. Prvé vydanie. Alfa, Bratislava, 1990. ISBN 80-05-00679-9.

[5] Jitka Dupačová, Petr Lachout. Úvod do optimalizace. Vydání první. Matfyzpress, Praha, 2011. ISBN 978-80-7378-176-7.

[6] Saul I. Gass. Lineárne programovanie metódy a aplikácie. Druhé prepracované vydanie. Alfa, Bratislava, 1972.

[7] M. S. Makower, E. Williamson. Základy operačnej analýzy (úlohy, technika, cvičenia). Prvé vydanie. Alfa, Bratislava, 1970.

[8] Jiří Matoušek, Jaroslav Nešetřil. Kapitoly z diskrétní matematiky. Dotisk druhého, opraveného vydání. Nakladatelství Karolinum, Praha, 2002. ISBN 80-246-0084-6.

Zoznam obrázkov