VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY

INSTITUTE OF MANUFACTURING TECHNOLOGY

SÍŤOVÁ SIMPLEXOVÁ METODA

NETWORK SIMPLEX METHOD

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — **LUKÁŠ KHÝR**

AUTHOR

VEDOUCÍ PRÁCE — **RNDr. PAVEL POPELA, Ph.D.**

SUPERVISOR

BRNO 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání bakalářské práce

Ústav:	Ústav matematiky
Student:	**Lukáš Khýr**
Studijní program:	Aplikované vědy v inženýrství
Studijní obor:	Matematické inženýrství
Vedoucí práce:	**RNDr. Pavel Popela, Ph.D.**
Akademický rok:	2017/18

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma bakalářské práce:

Síťová simplexová metoda

Stručná charakteristika problematiky úkolu:

Student se seznámí s problematikou optimalizačních modelů pro dopravní úlohy. Zaměří se zejména na lineární úlohy a speciálně na úlohy o toku v síti. Důraz bude klást na algoritmy řešení těchto úloh, a to zejména na simplexovou metodu a její síťovou modifikaci. Metoda bude implementována a testována na vybraných testovacích datech odvozených z reálných aplikací.

Cíle bakalářské práce:

1. Práce uvede přehled vybraných poznatků matematického modelování v oblasti optimalizace.
2. Přehledně budou zpracovány optimalizační síťové modely.
3. Autor podrobně popíše síťovou simplexovou metodu.
4. Práce zahrne původní autorův návrh implementace síťové simplexové metody v MATLABu.
5. Výsledná implementace bude testována pro vybrané soubory vstupních dat různé velikosti.

Seznam doporučené literatury:

GHIANI, Gianpaolo, Gilbert LAPORTE and Roberto MUSMANNO. Introduction to logistics systems planning and control. Hoboken, N.J.: John Wiley & Sons. 2004. ISBN 0-470-84917-7.

BAZARAA, Mokhtar S., John J. JARVIS and Hanif D. SHERALI. Linear programming and network flows. 4th ed. Hoboken, N.J.: John Wiley & Sons, 2010. ISBN 978-0-470-46272-0.

WOLSEY, Laurence A. Integer programming. New York: John Wiley & Sons, 1998. ISBN 978-0-4-1-28366-9.

NASH, Stephen and Ariela SOFER. Linear and nonlinear programming. McGraw-Hill, 1995. ISBN 978-0-89871-661-0.

WILLIAMS, H. Paul. Model building in mathematical programming. 5th ed. Hoboken, N.J.: John Wiley & Sons, 2013. ISBN 978-1-118-44333-0.

Termín odevzdání bakalářské práce je stanoven časovým plánem akademického roku 2017/18

V Brně, dne

L. S.

prof. RNDr. Josef Šlapal, CSc.
ředitel ústavu

doc. Ing. Jaroslav Katolický, Ph.D.
děkan fakulty

Abstrakt
Práce je zaměřena na shrnutí poznatků týkajících se matematického modelování v oblasti optimalizace. Budeme se podrobněji zabývat simplexovou metodou a především její síťovou modifikací, která má využití v různých praktických aplikacích. Pomocí implementace těchto dvou metod v Matlabu budou řešeny úlohy různé velikosti a budou porovnávána jejich řešení. V závěru práce se řeší úloha s reálnými daty poskytnutá Ústavem procesního inženýrství.

Summary
This thesis is focused on summary knowledges relating to mathematical modeling in optimization area. We will deal with simplex method in detail and especially its network modification, which is used in various practical applications. We will solve tasks of various sizes and compare their solutions using implementation these two methods in Matlab. There is solved a task with real data provided by the Institute of Process Engineering at the end of the work.

Klíčová slova
teorie grafů, optimalizace, lineární programování, simplexová metoda, síťová simplexová metoda

Keywords
graph theory, optimization, linear programing, simplex method, network simplex nethod

KHÝR, L. *Síťová simplexová metoda*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2018. 39 s. Vedoucí diplomové práce RNDr. Pavel Popela, Ph.D.

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

Lukáš Khýr

Rád bych poděkoval svému vedoucímu bakalářské práce, RNDr. Pavlu Popelovi, Ph.D., za cenné rady, věcné připomínky a vstřícnost při konzultacích a také Ing. Radovanu Šomplákovi, Ph.D., za vstřícnost při konzultacích a poskytování dat.

Lukáš Khýr

Obsah

1 Úvod **2**

2 Teorie grafů **3**
2.1 Základní typy grafů 3
2.2 Typy podgrafů 5

3 Optimalizace **8**
3.1 Úvod do optimalizace 8
3.2 Konvexnost 10
3.3 Simplexová metoda 12
3.4 Síťové úlohy 14

4 Síťová simplexová metoda **19**
4.1 Hlavní kroky 19
4.2 Fáze 1 19
4.3 Fáze 2 20

5 Srovnávání metod **32**
5.1 Testovací úlohy 32
5.2 Reálná data 33

6 Závěr **37**

7 Seznam příloh **39**

1. Úvod

Tato práce se zabývá síťovou simplexovou metodou. Text práce je rozložen do následujících kapitol.

V první kapitole se zabýváme teorií grafů. Definujeme základní pojmy a s návazností se dostáváme například ke kostře grafu, kterou síťová simplexová metoda využívá, nebo k toku v síti. Pomocí nástrojů teorie grafů lze řešit také různé optimalizační úlohy, například problém nejkratší cesty. Ovšem tím se zde podrobněji nezabýváme.

Ve druhé kapitole se dostáváme obecně do problémů optimalizace. Po nastínění struktury zápisu optimalizačního problému se věnujeme pojmům a vlastnostem optimalizačních problémů, které souvisí se síťovou simplexovou metodou. Řeším tedy například konvexnost či linearitu problému. Dále se pak zabýváme klasickou simplexovou metodou a její postupy demonstrujeme na konkrétním příkladu. V závěru této kapitoly rozebíráme úlohu optimalizačního problému na síti.

Ve třetí kapitole přichází na řadu samotná síťová simplexová metoda. Zde vysvětlujeme její principy řešení, které demonstrujeme na konkrétních příkladech a přikládáme příslušnou část kódu v programu Matlab, která daný princip řeší. Díky své efektivitě hraje tato metoda významnou roli při řešení síťových úloh. V dnešní době má tedy mnohé uplatnění nejen v dopravních úlohách ale také v jakýchkoliv úlohách, které lze převést na síťové, například model odrážející časová omezení při plánování projektu.

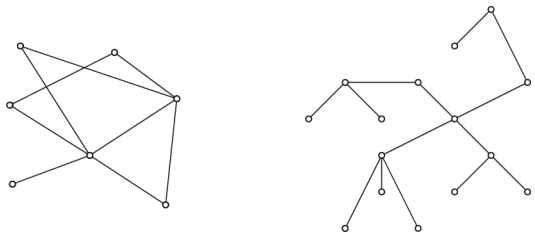
Poslední kapitola je zaměřena na řešení konkrétních příkladů a porovnání efektivity klasické a síťové metody. Pomocí reálných dat představující síť České republiky, poskytnutých Ústavem procesního inženýrství, byla vytvořena reálnoá síťová úloha a byla řešena již zmíněnými metodami. Pro výpočet klasickou simplexovou metodou byla použita funkce *linprog* implementovaná v programu Matlab a pro výpočet její síťovou modifikací byla využita mnou naprogramovaná metoda v programu Matlab.

2. Teorie grafů

V matematice se grafem obvykle rozumí znázornění funkční závislosti. Ovšem v teorii grafů se jím rozumí algebraická struktura, která popisuje objekty a vztahy mezi nimi. Graf si tedy můžeme představit jako objekty (vrcholy), které jsou propojené (hranami). Pokud je mezi dvěma vrcholy vazba, popisuje se jako dvojice (dvouprvková množina) příslušných vrcholů. [1]

2.1. Základní typy grafů

Definice 1 *Graf G (také jednoduchý graf nebo obyčejný graf) je uspořádaná dvojice G=(V,E), kde V je neprázdná množina vrcholů a E je množina hran – množina (některých) dvouprvkových podmnožin množiny V.*



Obrázek 2.1: Jednoduchý graf.

Vrcholy grafu se obvykle značí malými písmeny z konce abecedy (například $u, v, w, ...$). Hrany jsou dvouprvkové podmnožiny V, například $\{u, v\}$. Používá se však také zkrácený zápis uv (tedy uv je pouze zkrácený zápis $\{u, v\}$). Říká se, že „vrcholy u a v patří hraně uv," nebo že vrcholy u a v jsou spojené hranou uv. Pokud hrana $uv \in E(G)$ (kde $E(G)$ značí E pro $G = (V, E)$), pak u a v jsou sousední v grafu G. Jestliže množina E hranu uv neobsahuje, pak se zřejmě vrcholy u, v nazývají nesousední nebo také nezávislé. Vrcholy hrany uv se nazývají koncové vrcholy hrany uv.

Definice 2 *Orientovaným grafem rozumíme uspořádanou dvojici $G = (V, E)$, kde V je množina vrcholů a množina orientovaných hran je $E \subseteq V \times V$.*

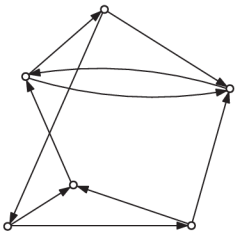
Orientovaná hrana uv již tedy není dvouprvková podmnožina $\{u, v\}$, ale uspořádaná dvojice (u, v). Vrchol u je počáteční a vrchol v koncový. Při kreslení se orientované hrany znázorňují šipkami (viz obrázek 2.2)

Definice 3 *Graf na n vrcholech, kde $n \in N$, který obsahuje všech $\binom{n}{2}$ hran se nazývá úplný nebo také kompletní graf a značí se K_n.*

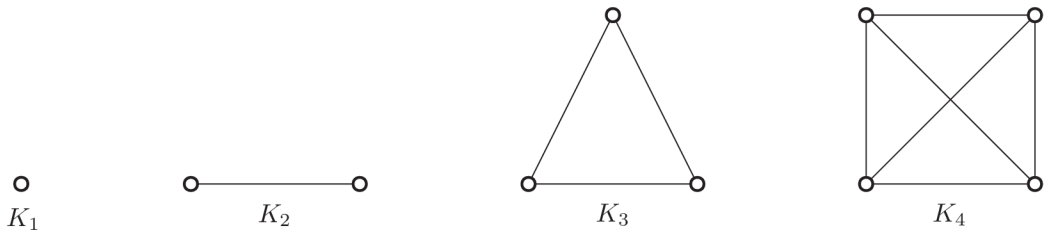
Kompletní graf K_n lze popsat následovně:

$$K_n = (V, E) : V = \{1, 2, ..., n\}, E = \{ij : i, j = 1, 2, ..., n \wedge i \neq j\}$$

[1] V této kapitole byly definice a věty převzaty z knihy [4]



Obrázek 2.2: Orientovaný graf.



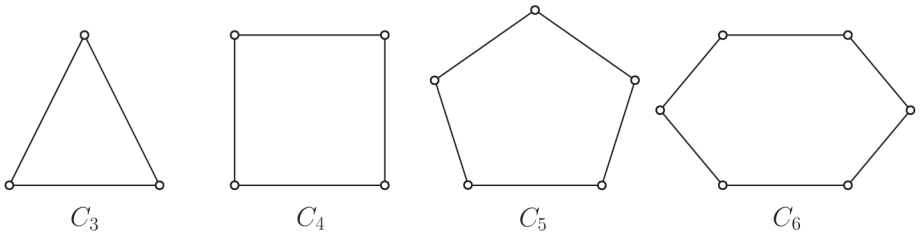
Obrázek 2.3: Příklady kompletních grafů pro malé n.

Definice 4 *Graf na n vrcholech (kde n = 3), které jsou spojeny po řadě n hranami tak, že každý vrchol je spojen s následujícím vrcholem a poslední vrchol je navíc spojen s prvním vrcholem, se nazývá cyklus na n vrcholech a značí se C_n. Číslo n je pak délka cyklu C_n.*

Cyklus C_n lze popsat následovně:

$$C_n = (V, E) : V = \{1, 2, ..., n\}, E = \{i(i+1) : i = 1, 2, ...n-1\} \cup \{1n\},$$

kde číslo n je alespoň 3. Jak již bylo zmíněno, zápis $i(i+1)$ je zkrácený zápis množiny $\{i, i+1\}$, a jak obrázek 2.4 napovídá, cyklu délky tři se někdy říká trojúhelník a cyklu délky čtyři čtverec. Lze si také povšimnout, že trojúhelník C_3 je současně kompletním grafem K_3.



Obrázek 2.4: Příklady cyklů pro malé n.

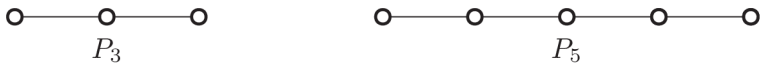
Definice 5 *Graf na n vrcholech, které jsou spojeny po řadě n − 1 hranami, se nazývá cesta a značí se P_n.*

Cestu P_n lze popsat následovně:

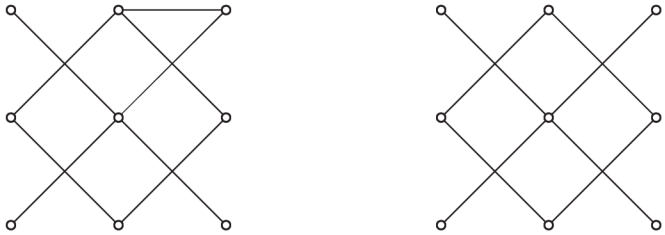
$$P_n = (V, E) : V = \{1, 2, ..., n\}, E = \{i(i+1) : i = 1, 2, ...n-1\}.$$

První a poslední vrchol cesty se nazývá koncový, ostatní vrcholy (pokud existují) jsou vnitřní vrcholy.

Definice 6 *Graf $G = (V, E)$ se nazývá souvislý, pokud $\forall u, v \in V$ existuje spojení (cesta).*



Obrázek 2.5: Příklad cest pro malé n.



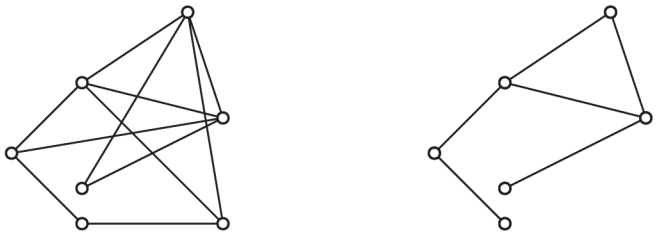
Obrázek 2.6: Příklad souvislého (vlevo) a nesouvislého (vpravo) grafu.

2.2. Typy podgrafů

Pokud je potřeba, tak se při řešení úloh vynechávají některé hrany nebo vrcholy, případně obojí. Dostáváme s tedy k pojmu *podgraf*.

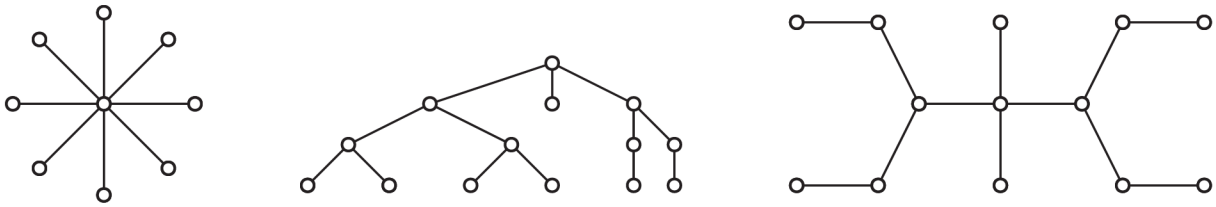
Definice 7 *Graf H nazveme podgrafem grafu G, jestliže $V(H) \subseteq V(G)$ a $E(H) \subseteq E(G)$. Píšeme $H \subseteq G$.*

Zřejmě pokud vynecháme některý vrchol, musíme také vynechat všechny hrany s ním spojené. Pro nazvání $H = (V(H), E(H))$ grafem, musí $E(H)$ obsahovat (některé) dvouprvkové podmnožiny $V(H)$. Podgraf H na obrázku 2.7 vznikl z grafu G odebráním jednoho vrcholu (a hran s ním spojených) a několika dalších hran.



Obrázek 2.7: Graf G (vlevo) a jeho podgraf H (vpravo).

Definice 8 *Strom je souvislý graf, který neobsahuje cyklus.*



Obrázek 2.8: Příklady různých stromů.

Věta 1 *Strom s n vrcholy má právě $n-1$ hran.*

Věta 2 *Každý strom skládající se alespoň z 2 uzlů má alespoň jeden konec (uzel ke kterému je připojena pouze jedna hrana).*

Věta 3 *Mezi každými dvěma vrcholy stromu existuje právě jedna cesta.*

Důsledek 1 *Přidáním jedné (nové) hrany do stromu (s alespoň třemi vrcholy) vznikne graf s právě jedním cyklem.*

Věta 4 *Strom je minimální souvislý graf (s daným počtem vrcholů).*

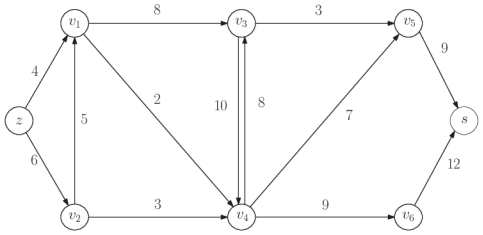
Definice 9 *Graf se nazývá ohodnocený, jestliže každé hraně přiřadíme reálné číslo (tzv. ohodnocení hrany nebo váha hrany), a (kladně) vážený, pokud je ohodnocení každé hrany kladné číslo.*

Definice 10 *Kostrou souvislého grafu G rozumíme takový podgraf, který obsahuje všechny vrcholy grafu G a který je stromem. Váhou kostry ohodnoceného grafu G rozumíme součet vah všech hran této kostry.*

Kostry jsou významné hlavně pro jejich minimalitu co do počtu hran, kdy je současně zachována souvislost grafu (viz věta 4). Zřejmě když je ohodnocení všech hran totožné, tak budou mít všechny kostry grafu stejnou váhu. Ovšem liší-li se ohodnocení jednotlivých hran, mohou mít různé kostry téhož grafu různou váhu.

Definice 11 *Síť je čtveřice $S = (G, z, s, w)$, kde G je orientovaný graf. Vrcholy $z \in V(G)$, $s \in V(G)$ budeme nazývat zdroj a stok v síti S. Funkce $w : E(G) \to R^+$ je kladné ohodnocení hran, které každé hraně přiřadí tzv. kapacitu hrany.*

Síť je vlastně orientovaný graf, ve kterém máme dva stěžejní vrcholy (zdroj a stok) a ohodnocené hrany (může být uvedena ještě jejich kapacita). Kapacita hrany může představovat šířku silnice, maximální počet aut, který projede za minutu, počet přenesených megabytů za vteřinu a podobně. Někdy se zjednodušeně mluví o síti G, přičemž zdroj, stok i kapacity hran jsou známy z kontextu. Příklad sítě je na obrázku 2.9.



Obrázek 2.9: Síť $S = (G, z, s, w)$.

Definice 12 *Tok v síti $S = (G, z, s, w)$ je funkce $f : E(G) \to R_0^+$, která má následující vlastnosti*

1. *ohodnocení hrany $f(e)$ se nazývá tok hranou a nesmí překročit kapacitu hrany*

$$\forall e \in E(G) : 0 \leq f(e) \leq w(e),$$

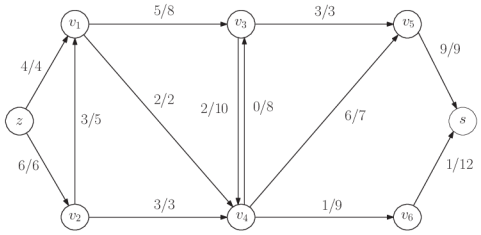
2. pro všechny vrcholy s výjimkou zdroje a stoku platí tzv. zákon kontinuity:

$$\forall v \in V(G), v \neq z, v \neq s : \sum_{e \to v} f(e) = \sum_{e \leftarrow v} f(e).$$

Pro zdroj obecně platí $\sum_{e \to v} f(e) \leq \sum_{e \leftarrow v} f(e)$*, zatímco pro stok platí opačná nerovnost* $\sum_{e \to v} f(e) \geq \sum_{e \leftarrow v} f(e)$*. Velikost toku* f *označíme* $\|f\|$ *a je dána vztahem*

$$\| f \| = \sum_{e \leftarrow z} f(e) - \sum_{e \to z} f(e)$$

Výraz $e \to v$, resp. $e \leftarrow v$ (v definici 12) označuje hranu e směřující k vrcholu v, resp. hranu e směřující od vrcholu v (podobně u $e \to z$, resp. $e \leftarrow z$). Na obrázku 2.10 vidíme příklady toků v nějaké síti. Čísla a/b přiřazená hranám mají význam toku a kapacity příslušné hrany. Definici 11, která předpokládá existenci jediného zdroje a jediného stoku, není obtížné zobecnit. Místo jediného zdroje z a jediného stoku s můžeme pracovat s množinou zdrojů $Z = \{z_1, z_2, ..., z_p\}$ a množinou stoků $S = \{s_1, s_2, ..., s_q\}$.



Obrázek 2.10: Příklad toku v síti (G, z, s, w).

3. Optimalizace

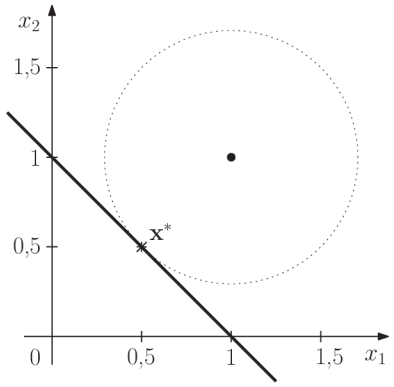
Optimalizační modely se snaží z matematického hlediska dosáhnout řešení problému nejlepší cestou. To může znamenat například provozování podniku, aby se maximalizoval zisk, minimalizovala ztráta, maximalizovala účinnost nebo minimalizovala rizika. Mohlo by to také znamenat navržení mostu při minimální hmotnosti nebo maximální pevnosti. Můžeme si pod tím představit také výběr letu pro letadlo, abychom minimalizovali dobu letu nebo spotřebu paliva. Touha řešit problém optimálním způsobem je tak běžná, že optimalizační modely vznikají téměř v každé oblasti použití. Dokonce byly použity při vysvětlení zákonů přírody, jako například ve Fermatově odvození zákona o lomu světla. [1]

3.1. Úvod do optimalizace

Podívejme se na jednoduchou úlohu nalezení bodu na přímce $x_1 + x_2 = 1$, který je nejblíže bodu $(1, 1)^T$ (viz obrázek 3.1). Tento problém můžeme formulovat takto:

$$\begin{aligned} &\text{minimalizuj} \quad f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 1)^2 \\ &\text{za podmínky} \quad x_1 + x_2 = 1. \end{aligned} \tag{3.1}$$

Optimálním bodem této úlohy je bod $(0,5; 0,5)^T$. Teto ilustrace je jednoduchým příkladem optimalizačního problému. V těchto problémech typicky minimalizujeme nebo maximalizujeme funkci f (účelová funkce) za nějakých předepsaných podmínek (ty určují množinu přípustných řešení S). Tyto podmínky jsou většinou definovány pomocí vhodných omezení proměnných (viz podmínka v úloze 3.1).



Obrázek 3.1: Nelineární optimalizace. Přípustná množina je zvýrazněná přímka.

V tomto případě byla naše funkce nelineární $f(\mathbf{x}) = (x_1 - 1)^2 + (x_2 - 1)^2$ a množina přípustných řešení S byla definována jedním lineárním omezením $x_1 + x_2 = 1$. Pro zkrácení zápisu budeme dále psát min (minimalizuj) a podm. (za podmínek). V následujícím příkladu si ukážeme, že množinu přípustných řešení S můžeme mít definovanou několika omezeními.

$$\min \quad f(\mathbf{x}) = x_1$$

[1] V této a následující kapitole vycházím z knih: [1], [2], [3], [5], [6].

$$\begin{aligned} \text{podm.} \quad & x_1^2 \leq x_2, \\ & x_1^2 + x_2^2 \leq 3. \end{aligned}$$

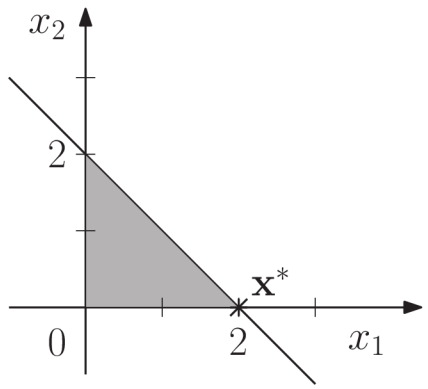
Stejně jako mohou existovat různá omezení, tak žádná omezení existovat nemusí, jak můžeme vidět v následujícím příkladu:

$$\min \quad f(\mathbf{x}) = (e^{x_1} - 2)^2 + (x_2 - 2)^2.$$

Množina přípustných řešení S je zde celý dvojrozměrný prostor $\mathbf{R}^2$.

Jak jsme viděli na příkladech, množina přípustných řešení S může být definovaná rovnostmi, nerovnostmi nebo omezení neuvažujeme. Funkce definující účelovou funkci může být lineární nebo nelineární. Příklady, které jsme dosud uvedli, jsou nelineární, protože vždy byla alespoň jedna z těchto částí modelu (účelová funkce nebo některá z podmínek přípustné množiny S) nelineární. Pokud jsou ovšem účelová funkce a všechna omezení lineární, pak se takový problém nazývá úloha lineární optimalizace nebo také lineární program. Jako příklad si uveďme následující problém.

$$\begin{aligned} \min \quad & f(\mathbf{x}) = 3x_1 + x_2 \\ \text{podm.} \quad & x_1 + x_2 \leq 2, \\ & x_1 \geq 0, x_2 \geq 0. \end{aligned}$$

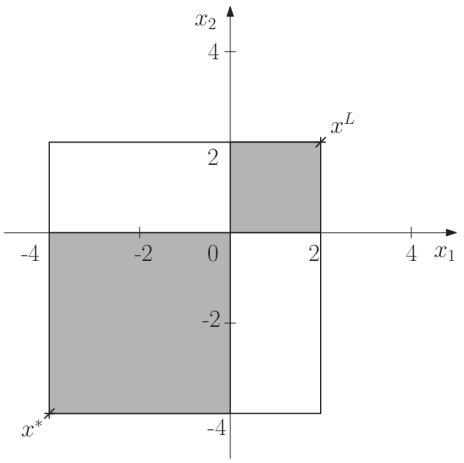


Obrázek 3.2: Lineární problém. Množina přípustných řešení je šedá.

Množinu přípustných řešení S můžeme vidět na obrázku 3.2. Optimálním řešením je $\mathbf{x}^* = (2, 0)^T$. Pohleďme ještě na poslední příklad:

$$\begin{aligned} \max \quad & f(\mathbf{x}) = (x_1 + x_2)^4 \\ \text{podm} \quad & x_1, x_2 \geq 0, \\ & -4 \leq x_1 \leq 2, \\ & -4 \leq x_2 \leq 2. \end{aligned}$$

Jak je vidět, jde o nelineární optimalizaci. Množina přípustných řešení S je nastíněna na obrázku 3.3. Pro bod $\mathbf{x}_L = (2, 2)^T$ je hodnota účelové funkce $f(\mathbf{x}_L) = 256$, což je větší hodnota funkce než v jakémkoli „blízkém“ okolí bodu v množině přípustných řešení S,



Obrázek 3.3: Lokální a globální řešení. Přípustná oblast je šedá.

proto je tento bod nazýván lokální optimum. Ale naopak bod $\mathbf{x}^* = (-4, -4)^T$ má hodnotu účelové funkce $f(\mathbf{x}^*) = 4096$, což je největší hodnota pro celou oblast možných řešení. Tento bod se nazývá globální optimum.

Ne vždy jsou tyto úlohy takto triviální, a proto nastává problém, že nevíme, zda jsme našli lokální nebo globální optimum. Jedinou výjimkou jsou problémy konvexní optimalizace, ve kterých platí, že lokální optimum je vždy globální. Lineární programy jsou konvexní, takže pro tuto skupinu problémů jsou lokální optima také globálními.

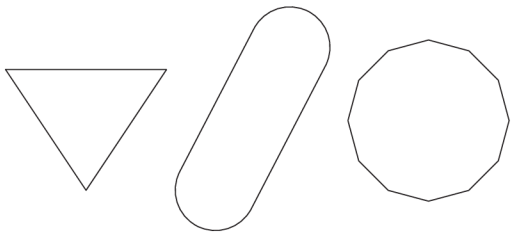
3.2. Konvexnost

Jak jsme již zmínili, vlastnost problému být konvexní hraje důležitou roli pro jeho řešení. Připomeňme, že pokud je problém konvexní, pak platí, že každé lokální optimum je zároveň i globálním optimem. Aby byl problém konvexní, musí být konvexní účelová funkce a také musí být konvexní množina přípustných řešení. V této kapitole se definuje, co přesně tato vlastnost znamená. Začněme s přípustnou množinou.

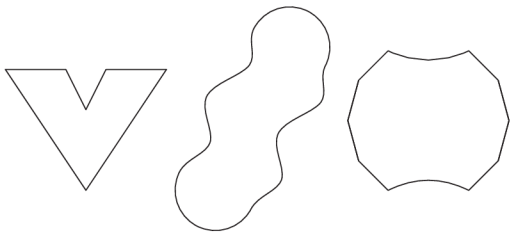
Definice 13 *Množina S je konvexní, pokud $\forall x, y \in S$ platí:*

$$\alpha x + (1 - \alpha)y \in S \ \forall \alpha : 0 \leq \alpha \leq 1.$$

Jinými slovy, pokud x a y leží v S, pak úsečka spojující x a y leží v S. Na obrázku 3.4 můžeme vidět příklady takové množiny. Obecněji se dá říci, že každá množina, která je definovaná systémem lineárních podmínek, je konvexní.



Obrázek 3.4: Konvexní množiny.



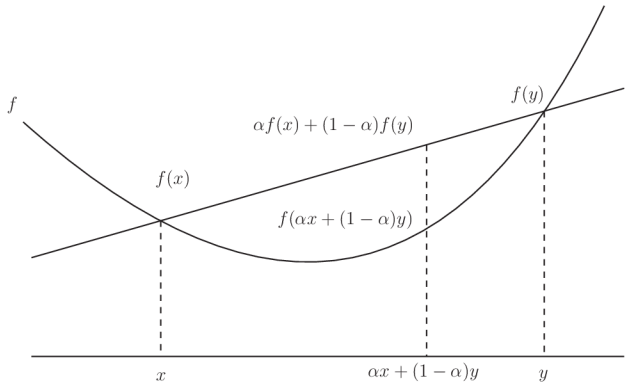
Obrázek 3.5: Nekovexní množiny.

Definice 14 *Funkce f je konvexní na konvexní množině S, pokud platí:*

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y)$$

$$\forall \alpha : 0 \leq \alpha \leq 1, \forall x, y \in S.$$

Tato definice nám říká že spojnice bodů $(x, f(x))$ a $(y, f(y))$ leží na grafu nebo nad grafem této funkce. Intuitivně řečeno graf funkce bude mít tvar „misky“ (viz obrázek 3.6).



Obrázek 3.6: Konvexní funkce (obrázek převzat z knihy [3]).

Definice 15 *Funkce f je konkávní na S pokud platí:*

$$f(\alpha x + (1-\alpha)y) \geq \alpha f(x) + (1-\alpha)f(y)$$

$$\forall\ \alpha, 0 \leq \alpha \leq 1, \forall x, y \in S.$$

Definice 16 *Funkce f je striktně kovexní pokud platí:*

$$f(\alpha x + (1-\alpha)y) < \alpha f(x) + (1-\alpha)f(y)$$

$$\alpha : 0 < \alpha < 1 \wedge \forall x \neq y,\ x, y \in S.$$

Problém konvexní optimalizace pak vypadá následovně:

$$\min \quad f(x)$$

$$\text{podm.} \quad x \in S,$$

kde množina S je konvexní a funkce f na množině S je také konvexní. Problém

$$\min \quad f(x)$$

$$\text{podm.} \quad g_i(x) \geq 0, i = 1, ..., m$$

je konvexní, pokud f je konvexní a funkce $\{g_i\}$ jsou konkávní.

Definice 17 *Nechť $\boldsymbol{x}^*$ je lokální minimum konvexního optimalizačního problému. Pak $\boldsymbol{x}^*$ je také globální minimum. Pokud je účelová funkce je striktně konvexní, pak $\boldsymbol{x}^*$ je jediné globální minimum.*

3.3. Simplexová metoda

Simplexová metoda je široce používaná metoda pro problémy lineárního programování. Řeší pouze lineární problémy, ovšem techniky, které používá, se dají aplikovat i na nelineární úlohy (viz [1]). Metoda postupuje od základního řešení, v každém svém kroku řešení pozmění takovým způsobem, aby hodnota účelové funkce byla vyšší než v předchozím kroku. Přesný popis algoritmu je např. v [3], zde ji vysvětlíme na příkladu. Pro řešení pomocí S.M. (simplexové metody) byla v této práci použita funkce *linprog* v programu Matlab.

Základní postup si vysvětleme na následujícím příkladu

$$\begin{aligned} \min \quad & -5x_1 - 4x_2 \\ \text{podm.} \quad & x_1 + 2x_2 \leq 6, \\ & 2x_1 - x_2 \leq 4, \\ & 5x_1 + 3x_2 \leq 15, \\ & x_1, x_2 \leq 0. \end{aligned} \tag{3.2}$$

Pro řešení převedeme úlohu na maximalizační (tj. využít $\min f(\mathbf{x}) = \max -f(\mathbf{x})$) a omezení převedeme na standardní tvar a to tak, že podmínky typu $\leq$ přepíšeme na $=$ a k levé straně přičteme novou proměnnou. Podmínky typu $\geq$ přepíšeme na $=$ a od levé strany odečteme novou nezápornou proměnnou. Tím získáme následující úlohu ve standardním tvaru:

$$\begin{aligned} \max \quad & 5x_1 + 4x_2 \\ \text{podm.} \quad & x_1 + 2x_2 + x_3 = 6, \\ & 2x_1 - x_2 + x_4 = 4, \\ & 5x_1 + 3x_2 + x_5 = 15, \\ & x_1, x_2, x_3, x_4, x_5 \geq 0. \end{aligned}$$

Pro přehlednost využijeme tabulkového zápisu.

x_1	x_2	x_3	x_4	x_5	b
1	2	1	0	0	6
2	-1	0	1	0	4
5	3	0	0	1	15
5	4	0	0	0	0

Tabulka 3.1: Iterace 1.

Sloupce v tabulce 3.1 odpovídají jednotlivým proměnným, řádky jednotlivým podmínkám. V posledním řádku je zapsáno kritérium ve tvaru $0 = (5x_1 + 4x_2 + 0x_3 + 0x_4 + 0x_5) - z$. V posledním řádku sloupce b je hodnota účelové funkce za předpokladu, že jsou všechny nebázové proměnné nulové (ty, které nemají ve svých sloupcích jednotkovou matici - x_1, x_2). Jak je vidět, nejprve se tato jednotková matice nachází u přidaných proměnných x_3, x_4, x_5 (v tabulce 3.1 podbarveno žlutě). Tedy funkce vypadá následovně:

$$z = f(\mathbf{x}) = 5 \cdot 0 + 4 \cdot 0 + 0 \cdot 6 + 0 \cdot 4 + 0 \cdot 15 = 0.$$

Nyní zvolíme hlavní sloupec. To je sloupec, na kterém hodnota funkce může při jednotkové změně proměnné nejvíce růst. Pro naši funkci je to zřejmě x_1, jelikož má tato proměnná hodnotu koeficientu nejvyšší (5). Nyní hledáme hlavní prvek (v simplexových tabulkách podbarven červeně), a to tak, že v hlavním sloupci tabulky 3.1 hledáme takovou kladnou hodnotu, jejíž podíl s odpovídající hodnotou ve sloupci b je minimální. Pro náš případ schématicky popíšeme: $6/1 = 6, 4/2 = 2, 15/5 = 3$. Hlavním prvkem je tedy hodnota v prvním sloupci a druhém řádku. Následuje přepočítání simplexové tabulky. Postupujeme tak, aby z báze vypadl sloupec obsahující jedničku na řádku hlavního prvku a nahradil jej v bázi hlavní sloupec. Koeficienty v posledním řádku pod bází musí být nulové. Nejprve se tedy vydělí hlavní řádek hlavním prvkem (na místě hlavního prvku vznikne jednička) a následně opačnou hodnotou prvku na hlavním sloupci vynásobíme hlavní řádek a přičteme jej k řádku s prvkem, kterým jsme násobili (na místě násobícího prvku vznikne nula). Pro náš případ popíšeme postup schématicky: řádek(2)/2 $\rightarrow$ (řádek(2) $\cdot (-1)$)+řádek(1) $\rightarrow$ (řádek(2) $\cdot (-5)$)+řádek(3) $\rightarrow$ (řádek(2) $\cdot (-5)$)+řádek(4). V tabulce 3.2 je vidět nově vzniklá simplexová tabulka.

x_1	x_2	x_3	x_4	x_5	b
0	5/2	1	-1/2	0	4
1	-1/2	0	1/2	0	2
0	11/2	0	-5/2	1	5
0	13/2	0	-5/2	0	-10

Tabulka 3.2: Iterace 2.

Další postup je obdobný jako v předchozím případě tabulky 3.1. Hlavní prvek je 11/2. Tabulku, která následuje je vidět v tabulce 3.3. Hlavním prvkem je nyní 7/11.

x_1	x_2	x_3	x_4	x_5	b
0	0	1	7/11	-5/11	19/11
1	0	0	3/11	1/11	27/11
0	1	0	-5/11	2/11	10/11
0	0	0	5/11	-13/11	-175/11

Tabulka 3.3: Iterace 3.

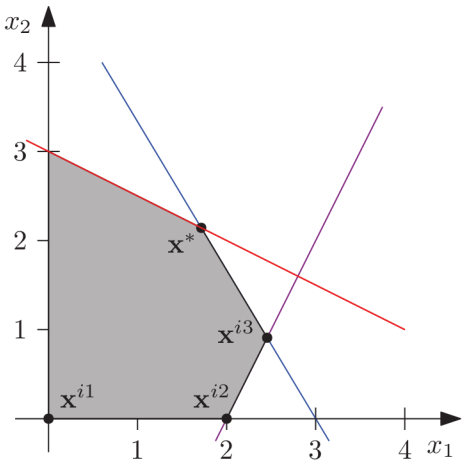
V tabulce 3.4 jsme se již dostali ke konečnému řešení, protože všechny koeficienty v posledním řádku u nebázových proměnných x_3, x_5 jsou záporné. V pravém dolním rohu tabulky 3.4 se nachází opačná hodnota z.

x_1	x_2	x_3	x_4	x_5	b
0	0	11/7	1	-5/7	19/7
1	0	-3/7	0	2/7	12/7
0	1	5/7	0	-1/7	15/7
0	0	-5/7	0	-6/7	-120/7

Tabulka 3.4: Iterace 4.

Nalezli jsme tedy jediné optimální řešení, konkrétně: $x_1 = 12/7, x_2 = 15/7, x_3 = 0$, $x_4 = 19/7, x_5 = 0$ a $z = 120/7$ ($z = 5 \cdot 12/7 + 4 \cdot 15/7$). Na obrázku 3.7 je naznačeno grafické řešení tého úlohy. Barevné přímky ilustrují hranici pro podmínky (viz 3.2, první podmínka - červená, druhá - fialová, třetí - modrá), šedá oblast je množina přípustných řešení, vektory $\mathbf{x}^{i1}$, $\mathbf{x}^{i2}$, $\mathbf{x}^{i3}$ označují řešení v iteracích 1, 2, 3 a vektor $\mathbf{x}^*$ je optimální řešení

vypočtené v iteraci 4. Lze si povšimnout, že řešení v jednotlivých iteracích odpovídá vždy rohu pětiúhelníku vytvořeného zadanými podmínkami (viz 3.2).



Obrázek 3.7: Ilustrace grafického řešení.

Obecně mohou nastat 3 speciální případy:

1. Zacyklení

   pro vyhnutí se zacyklení se používá například Blandovo pravidlo (více o Blandovu pravidlu lze nalézt například v[1]).

2. Nekonečně mnoho řešení

   nastává, pokud se v posledním řádku simplexové tabulky objeví nula na místě nebázové proměnné.

3. Neomezené řešení (tj, účelová funkce není na množině přípustných řešení maximalizační úlohy shora ohraničená)

   nastává, pokud se v hlavním sloupci objevují pouze záporné hodnoty (neuvažujeme poslední sloupec).

3.4. Síťové úlohy

Síťové problémy patří mezi úlohy lineárního programování. Tyto úlohy jsou definovány na síti, díky čemuž tento problém získává speciální vlastnosti, jenž lze využít při řešení. Síť může být fyzická, například silniční síť nebo síť telefonních linek, nebo může být pouze modelem, který odráží například časová omezení při plánování projektu.

Klasická síťová úloha se v maticovém tvaru zapíše následujícím způsobem:

$$\begin{aligned} \min \quad & z = \mathbf{c}^T\mathbf{x} \\ \text{podm.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}, \end{aligned}$$

kde z je hodnota funkce, kterou chceme minimalizovat, $\mathbf{c}$ je vektor jednotlivých cen jednotlivých hran, $\mathbf{x}$ je proměnná, $\mathbf{b}$ je vektor hodnot kapacit a poptávek uzlů, $\mathbf{l}$ určuje dolní meze, $\mathbf{u}$ horní meze, $\mathbf{A}$ je matice popisující systém sítě. Dolní meze bývají většinou nulové, ovšem není tomu tak vždy. Určitou úpravou úlohy lze tato čísla převést na nuly. Stačí zaměnit $\mathbf{l} \leq \mathbf{x}$ za $0 \leq \mathbf{x} - \mathbf{l} = \mathbf{x}^*$. Pokud je to nutné, můžeme eliminovat i horní hranici, stačí naši omezenou proměnnou nahradit dvěma proměnnými, které jsou nezáporné a neomezené: tedy omezení typu $x_j \leq u_j$ může být zaměněno za $x_j^+ - x_j^- \leq u_j$, kde x_j^+ a x_j^- jsou naše nové proměnné. V modelu musíme samozřejmě x_j nahradit $x_j^+ - x_j^-$. Tedy bez ztrát na obecnosti (a s pár úpravami modelu) bychom mohli dolní a spodní meze zanedbat.

To, co dělá síťové úlohy tak speciálními, jsou vlastnosti $\mathbf{A}$ a $\mathbf{b}$. Matice $\mathbf{A}$ má m řádků (každý řádek reprezentuje uzel) a n sloupců (každý sloupec reprezentuje hranu mezi dvěma uzly). Tato matice má speciální formu a to tu, že její prvky v řádku nabývají hodnoty 1 (pro hranu směřující od tohoto uzlu), -1 (pro hranu směřující k tomuto uzlu) a 0 (pro ostatní hrany). Její sloupce jsou ještě speciálnější, v každém sloupci matice se vyskytují pouze 3 hodnoty a to 1 (pro startovní uzel), -1 (pro cílový uzel) a 0 pro ostatní uzly. Zatímco hodnoty 1 a -1 se v každém sloupci vyskytují právě jednou, hodnot 0 je zde $m-2$. Součet řádků je zřejmě $\mathbf{0}$, takže jsou lineárně závislé. To znamená, že maximální hodnost matice $\mathbf{A}$ je $m - 1$. Předpokládejme $n \geq m$, pak všechny propojené hrany nepředstavují strom (viz věta 1).

Uzel s $b_i > 0$ představuje zásobu, s $b_i < 0$ představuje poptávku, s $b_i = 0$ pak překládku. Pokud

$$\sum_{i=1}^{m} b_i \neq 0, \tag{3.3}$$

pak úlohu nazýváme nevyváženou a před přistoupení k algoritmu musíme tuto úlohu vyvážit. Nechť S je celková zásoba a D je celková poptávka, pak

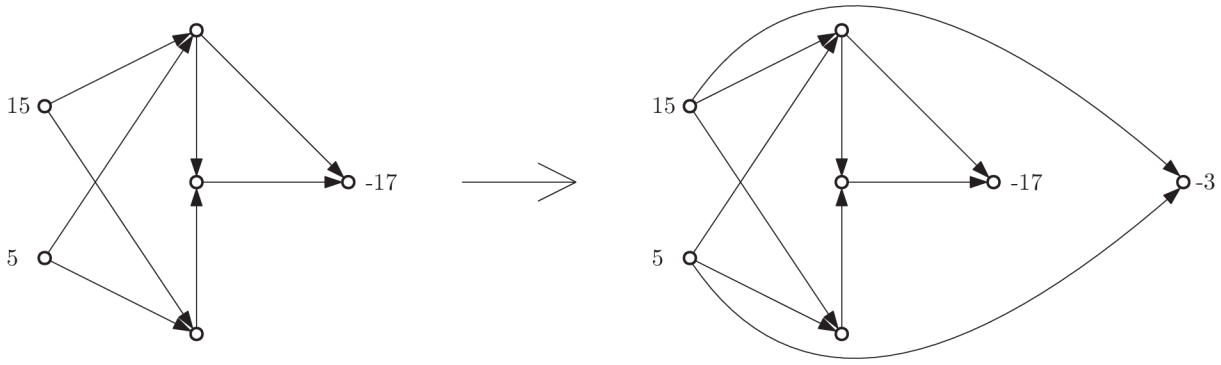
$$S = \sum_{i=1,\, b_i>0}^{m} b_i, \qquad D = \sum_{i=1,\, b_i<0}^{m} |b_i|. \tag{3.4}$$

Zřejmě, pokud $S = D$, je úloha vyvážená, ovšem pokud $S \neq D$, pak je nutné úlohu před jejím řešením vyvážit. Pokud je tedy

$S > D$ tj. zásoby jsou větší než celková poptávka, pak přidáme uzel s hodnotou $S - D$ a vytvoříme hranu mezi všemi uzly s $b_i > 0$ a námi přidaným uzlem, který představuje celkové zásoby. Horní mez (pokud je potřeba) nastavíme jako hodnotu b_s, kde $b_s = S - D$. Ceny pro nové hrany pak představují ceny nadměrné produkce,

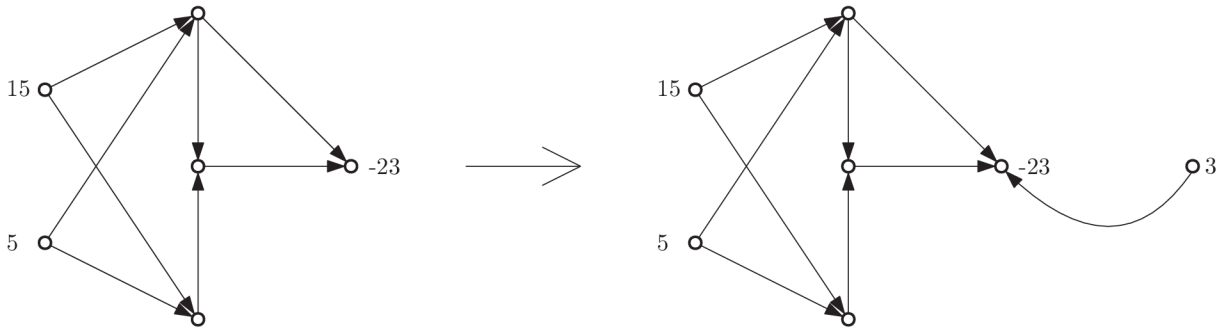
$S < D$ tj. poptávky jsou větší než celková zásoba, pak přidáme uzel s hodnotou $D - S$ a vytvoříme hranu mezi všemi uzly s $b_i < 0$ a námi přidaným uzlem. Horní mez (pokud je potřeba) nastavíme jako hodnotu $|b_s|$, kde $b_s = S - D$. Hodnota cen pro nové hrany pak představuje uniklý zisk.

Tímto jsme úlohu vyvážili a pro další optimalizaci není rozdíl mezi původními a námi přidanými hranami.



Obrázek 3.8: Vyvažování pro $S > D$.

Pro síť na obrázku 3.8 je $S = 20$ a $D = 17$, tedy $S > D$. Vytváříme tedy nový uzel s hodnotou $b_i = -3$ a nové hrany směřující z uzlů pro které platí nerovnost $b_i > 0$ k vytvořenému uzlu. Cenám nových hran je přiřazena hodnota v závislosti na konkrétní úloze a případná kapacita těchto hran se nastaví na hodnotu 3.



Obrázek 3.9: Vyvažování pro $S < D$.

Pro síť na obrázku 3.9 je $S = 20$ a $D = 23$, tedy $S < D$. Vytváříme tedy nový uzel s hodnotou $b_i = 3$ a nové hrany směřující z přidaného uzlu k uzlům pro které platí nerovnost $b_i < 0$ (na obrázku 3.9 je takový uzel jediný, proto pouze jedna hrana). Cenám nových hran je přiřazena hodnota v závislosti na konkrétní úloze a případná kapacita těchto hran se nastaví na hodnotu 3.

Než budeme pokračovat konkrétní úlohou, definujme si následující pojmy:

Definice 18 *Matice* $\boldsymbol{A}$ *struktury* $m \times n$ *se označuje jako dolní trojuhelníková matice, pokud* $a_{ij} = 0$ *pro* $j > i$, *kde* $i = 1, \ldots, m$ *a* $j = 1, \ldots, n$.

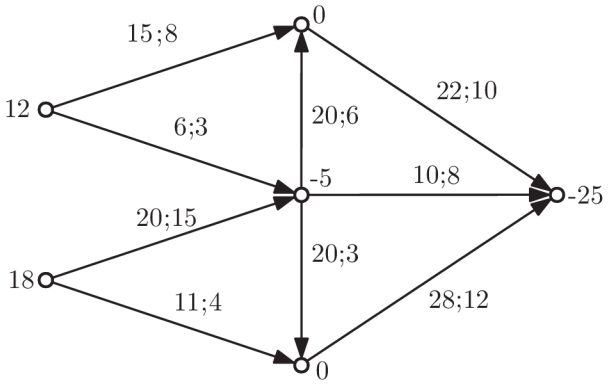
Následující matice **F**,**G** i **H** jsou dolní trojuhelníkové matice:

$$\mathbf{F} = \begin{pmatrix} 8 & 0 & 0 \\ 1 & 6 & 0 \\ 9 & 3 & 2 \end{pmatrix}, \ \mathbf{G} = \begin{pmatrix} 1 & 0 & 0 \\ 4 & 2 & 0 \\ 5 & 9 & 3 \\ 8 & 1 & 6 \end{pmatrix}, \ \mathbf{H} = \begin{pmatrix} 6 & 0 & 0 \\ 0 & 0 & 0 \\ 2 & 0 & 0 \\ 4 & 1 & 9 \end{pmatrix}.$$

Definice 19 *Hodnost matice* $\boldsymbol{A}$ *struktury* $m \times n$ *je rovna počtu lineárně nezávislých řádků (sloupců), značí se* $h(\boldsymbol{A})$.

Poznamenejm, že maximální $h(\mathbf{A}) = min(m, n)$.

Uvažujme dále následující síť:



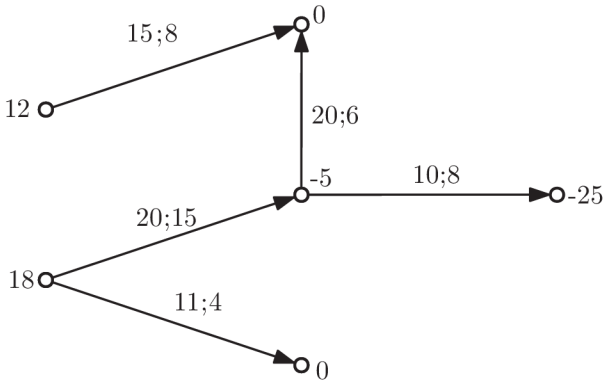
Obrázek 3.10: Daná síť.

a řešme

$$\begin{aligned} \min \quad & z = \mathbf{c}^T\mathbf{x} \\ \text{podm.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b}, \\ & \mathbf{l} \le \mathbf{x} \le \mathbf{u}, \end{aligned}$$

kde hodnoty v obrázku 3.10 u uzlů obsahuje vektor $\mathbf{b}$. Pro ohodnocení hran (g;h) platí, že hodnota g představuje kapacitu hrany a h její cenu. Zpracováním této sítě (obrázek 3.10) získáme následující vektory: $\mathbf{b}^T = (12, 18, 0, -5, 0, -25)$; $\mathbf{c}^T = (8, 3, 15, 4, 10, 6, 3, 8, 12)$; $\mathbf{u}^T = (15, 6, 20, 11, 22, 20, 20, 10, 28)$; $\mathbf{l}^T = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$ a matici $\mathbf{A}$:

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & -1 & -1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 \end{pmatrix}.$$



Obrázek 3.11: Kostra grafu z obrázku 3.10.

Tento příklad ilustruje problém minimalizace toku sítí, který nepřekročí kapacity hran, využije zásoby a splní poptávky. Každá submatice $\mathbf{B}$ matice $\mathbf{A}$, která odpovídá kostře grafu, má speciální strukturu, m řádků a $m-1$ sloupců (kostra grafu má tedy m uzlů

a $m-1$ hran). Díky tomu ji lze vhodnými úpravami přeskládat na dolní trojúhelníkovou matici s plnou hodností. Vezměme si síť z obrázku 3.10 a její matici **A** uvedenou výše. Potom submatice matice **A** odpovídající kostře grafu z obrázku 3.11 vypadá následovně:

$$\mathbf{B} = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccc} 12 & 24 & 25 & 43 & 46 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \end{array}.$$

Jak již bylo zmíněno, řádky matice **B** představují uzly, sloupce hrany a hodnoty v matici odpovídají tomu, odkud kam hrana směřuje (pokud hrana není, je v matici 0). Čísla před závorkou matice **B** představují očíslování uzlů. Čísla gh nad maticí **B** představují hrany (pro dané sloupce matice **A**), jejichž směr je od uzlu g k uzlu h. Pomocí vhodného přeskládání sloupců (1,3,2,4,5) a řádků (1,5,2,3,4,6) jsme matici **B** transformovali na matici

$$\hat{\mathbf{B}} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 \\ 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix},$$

která je, jak jsme již nastínili, dolní trojuhelníková a má plnou hodnost. Následující věta nám říká, že je to možné udělat vždy.

Věta 5 *Nechť **B** je submatice matice omezeních **A** odpovídající kostře grafu s m uzly. Pak **B** může být upravena na trojúhelníkovou matici s plnou hodností dimenze $m \times (m-1)$ s hodnotami ±1 na diagonále určené indexi i=j, i=1,…,m-1.*

Princip této transformace je jednoduchý, ale pokud by z našeho příkladu nebyl jasný, ukažme si algoritmus na vytvoření matice $\hat{\mathbf{B}}$. Mějme tedy kostru grafu a k ní odpovídající matici B. Vzhledem k větě 2 existuje uzel i, který je koncem kostry grafu, tudíž je připojen pouze k jedné hraně v kostře. Tedy i-tý řádek matice **B** má pouze jeden nenulový vstup s hodnotou ±1. Nechť je tato hodnota v j-tém sloupci. Nyní dáme řádek i před řádek 1 a zaměníme sloupec 1 a j. Tím jsme získali matici

$$\hat{\mathbf{B}}_1 = \begin{pmatrix} \pm 1 & \mathbf{0} \\ \mathbf{v}_1 & \mathbf{B}_1 \end{pmatrix},$$

kde $\mathbf{B}_1$ je submatice odpovídající kostře grafu s odebraným i-tým uzlem a $\mathbf{v}_1$ je sloupcový vektor obsahující nuly a jednu $+1$ nebo -1. Tento postup opakujeme pro matici $\mathbf{B}_1$, dostaneme $\hat{\mathbf{B}}_2$ pomocí $\mathbf{B}_2$ a $\mathbf{v}_2$ a dále pokračujeme dokud nezískáme matici $\hat{\mathbf{B}}$. Příslušný důkaz lze najít například v [3]. Platí tedy, že tato matice může být přeskládána na dolní trojuhelníkovou matici s ±1 na diagonále. Tyto poznatky využijeme u síťové simplexové metody v další kapitole.

4. Síťová simplexová metoda

Síťová simplexová metoda vychází ze simplexové metody. Její hlavní odlišností (od S.M.) je využití podbází (koster) a také nedochází k zacyklení (degradaci) problému. Používá se pro řešení úloh lineárního programování na sítích a díky její efektivitě u těchto úloh se stala důležitým nástrojem pro řešení síťových úloh, viz [3]. Tato metoda se zkládá ze dvou fází. Každá fáze se pak skládá ze 3 hlavních kroků.

4.1. Hlavní kroky

1. krok: Test optimality

   - nejdříve napočítáme multiplikátory y_i, a to následujícím způsobem: multiplikátor odpovídající koncovému uzlu kostry grafu nastavíme na nulu. Následně vypočítáme hodnoty y_i pomocí již známých y_j pro zbytek uzlů pomocí formule $y_i - y_j = c_{i,j}$, ($c_{i,j}$ jsou prvky vektoru cen).
   - poté napočítáme redukované ceny $d_{i,j}$: Pro všechny nebázické hrany (i, j) použijeme formuli $d_{i,j} = c_{i,j} - y_i + y_j$. Pokud platí, že $d_{i,j} \geq 0$ pro všechny nebázické hrany, pak je naše báze optimální.

2. krok: Pivotování

   - identifikujeme cyklus, který vznikne přidáním hrany (s, t) (pro tuto hranu je $d_{s,t}$ nejmenší) do kostry grafu. Následně zjistíme, jak moc můžeme zvýšit tok přidanou hranou, než se tok jinou hranou v cyklu vynuluje nebo dosáhne maximálního přípustného toku. Pokud dojdeme k maximálnímu toku hrany (s, t) (tzn. do toku odpovídajícímu $u_{s,t}$) a tok jinou hranou (i, j) se nevynuluje, ani nedosáhne svého maxima, pak tato hrana (s, t) nevchází do báze, ale naplníme ji (musí se upravit vektor **b**), a už ji dále neuvažujeme. Pokud to nastane, tak 3. krok přeskočíme a provedeme krok 1.

3. krok: Aktualizace

   - provedeme aktualizaci kostry grafu přidáním hrany (s,t) a odebráním hrany, jejíž tok se v kroku 2 dostal na hodnotu nula jako první a nebo se dostal jako první na své maximum. Pokud se dostal na své maximum, pak hrana odchází z báze naplněná a dále už ji neuvažujeme (musí se upravit vektor **b**).

4.2. Fáze 1

Ve fázi 1 inicializujeme základní přípustné řešení. Mějme tedy libovolnou síť. První fázi provedeme tak, že

a) přidáme k naší síti další uzel a vytvoříme nové hrany z ostatních uzlů k tomu našemu novému a to tak, že

   pokud $b_i > 0$,

pak vytváříme hranu z uzlu b_i do přidaného uzlu b_p a $l_{i,p} = 0$ a $u_{i,p} = b_i$,

pokud $b_i \leq 0$,

pak vytváříme hranu z přidaného uzlu b_p do uzlu b_i a $l_{p,i} = 0$ a $u_{p,i} = |b_i|$.

Nyní ještě nastavme cenu všech původních hran na 0 a cenu všech námi přidaných hran na 1.

b) alternativou k a) je nepřidávát nový uzel, ale jako uzel ke kterému vytváříme hrany vzít libovolný uzel sítě. Vytváření hran a nastavení cen probíhá stejně jako v první možnosti.

Jako kostru grafu teď bereme pouze námi přidané hrany. Nyní provádíme hlavní kroky síťové simplexové metody (viz 4.1.) s tím, že vždy když z báze odchází námi přidaná hrana, tak tuto hranu zrušíme, a už ji dále neuvažujeme. Po fázi 1 tedy získáme kostru grafu složenou jen z původních hran. Pokud nám v kostře zůstane námi přidaná hrana (pokud na začátku fáze 1 přidáváme nový uzel, pak nám nakonec vždy alespoň jedna námi přidaná hrana v bázi zůstane) a tok je pro tuto hranu (i, j) nenulový ($x_{i,j} \neq 0$), potom tato úloha nemá řešení (pomocí původních hran nelze vytvořit takový tok, aby jím proteklo vše, co potřebujeme). Pokud je tok pro tuto hranu (i, j) nulový ($x_{i,j} = 0$), potom se nic neděje (dostatečný tok jsme získali, aniž bychom museli použít přidanou hranu) - tato hrana je sice inicializovaná, ale není využita. Aby tomu tak zůstalo, položíme $c_{i,j} = 0$ a $u_{i,j} = 0$. Tím jsme získali základní přípustné řešení a můžeme pokračovat fází 2.

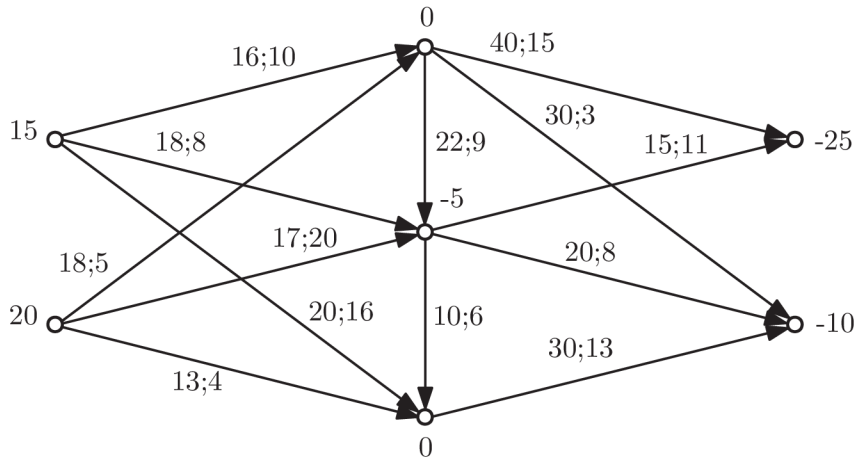
4.3. Fáze 2

Nyní vraťme původním hranám jejich ceny a, jak jsme již zmínili, upravme hrany přidané ve fázi 1, které nám zůstaly v bázi ($c_{i,j} = 0$ a $u_{i,j} = 0$ pro všechny přidané hrany). Nyní provádíme zmíněné hlavní kroky síťové simplexové metody (viz 4.1.) dokud nezískáme optimální řešení.

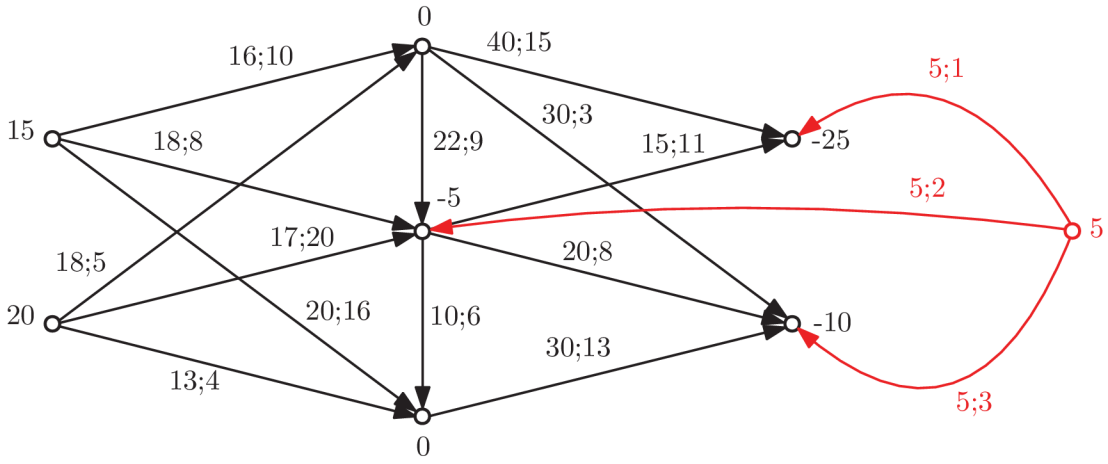
Vezměme si nyní následující síť (viz obrázek 4.1) a demonstrujme na ní síťovou simplexovou metodu. Potom vektory vypadají následovně: $\mathbf{b}^T = (15, 20, 0, -5, 0, -25, -10)$; $\mathbf{c}^T = (8, 16, 5, 20, 4, 9, 15, 3, 6, 11, 8, 13)$; $\mathbf{u}^T = (16, 18, 20, 18, 17, 13, 22, 40, 30, 10, 15, 20, 30)$ a matice

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & -1 & 0 & -1 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & -1 & -1 \end{pmatrix}.$$

Protože $\sum_{i=0} b_i \neq 0$, musíme úlohu nejprve vyvážit (viz obrázek 4.2). V obrázku je nově přidaný uzel, hrany a jejich ohodnocení červené.
Příslušný kód v Matlabu pro vyvažování úlohy:



Obrázek 4.1: Uvažovaná síť

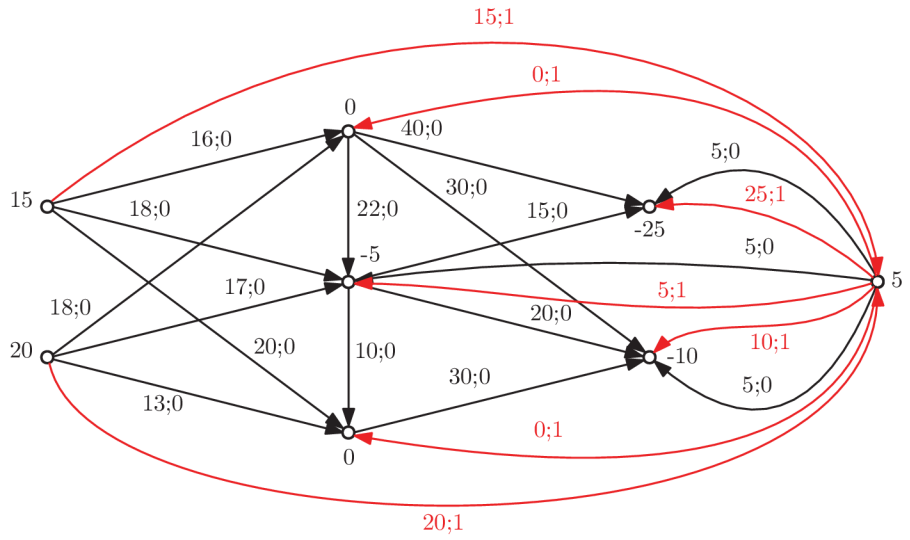


Obrázek 4.2: Vyvažování úlohy.

```
% vyhodnocení hodnot S a D
for i=1:length(b)
    if b(i)>0
        S=S+b(i);
        index_S(end+1)=(i);
    elseif b(i)<0
        D=D+abs(b(i));
        index_D(end+1)=(i);
    end
end
% nadmerna_produkce/odber - tím je myšlena cena přidaných hran.
% V programu defaultně nastaveno ale v případě konkrétních dat se to
upraví
nadmerna_produkce=ones(1,length(index_S))
nadmerny_odber=ones(1,length(b))
% vytváření nových sloupců matice A, cen a kapacit pro přidané hrany
v závislosti na velikosti S a D
```

```
if S>D
    b(end+1)=-(S-D)
    for i=1:length(index_S)
        A(index_S(i),end+1)=1;
        A(mA+1,end)=-1;
        index_n(nA+i,:)=[i mA+1];
        c(end+1)=nadmerna_produkce(i);
        u(end+1)=min(abs(b(end)),b(index_S(i)));
    end
elseif S<D
    b(end+1)=D-S ;
    for i=1:length(index_D)
        A(index_D(i),end+1)=-1;
        A(mA+1,end)=1;
        index_n(nA+i,:)=[mA+1 i];
        c(end+1)=nadmerny_odber(i)
        u(end+1)=min(b(end),abs(b(index_D(i))));
    end
end
```

Nyní pojďme na fázi 1, tedy přidání hran s úpravou cen (viz obrázek 4.3). V obrázku jsou nově přidané hrany a jejich ohodnocení červeně.



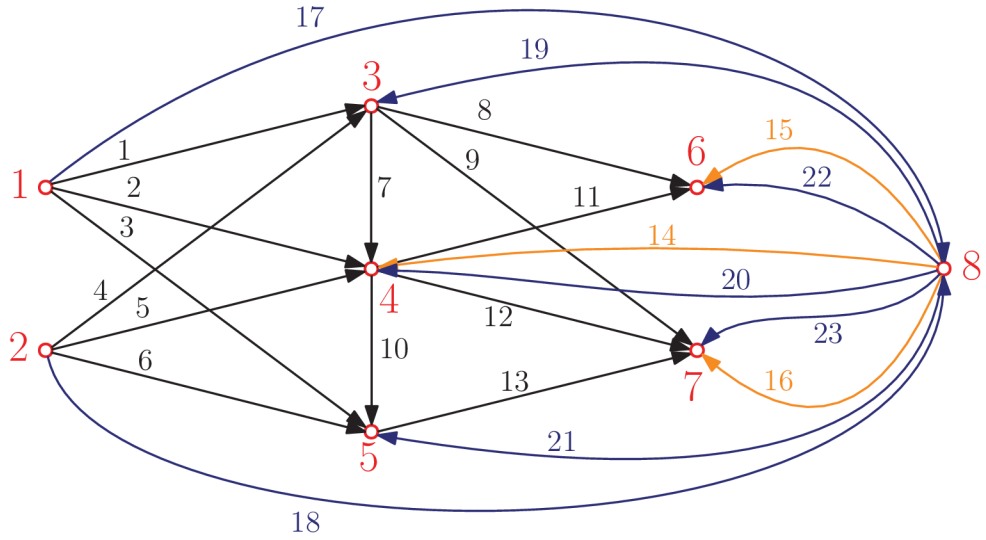
Obrázek 4.3: Přidání umělých hran, úprava cen.

Implementace přidávání umělých hran a úpravy cen v Matlabu:

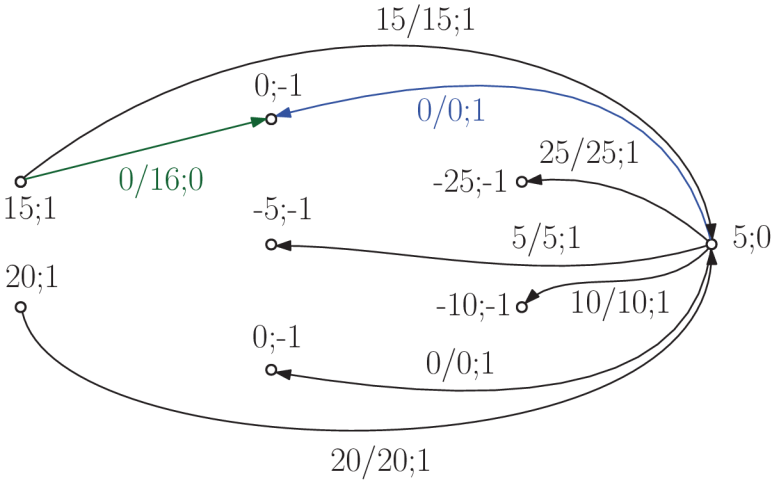
```
%% Přidání umělých hran
for i=1:length(b)-1
    if b(i)>0
        A(i,nA+i)=1;
        index_n(nA+i,1)=i;
        A(length(b),nA+i)=-1;
```

```
        index_n(nA+i,2)=length(b);
    else
        A(i,nA+i)=-1;
        index_n(nA+i,2)=i;
        A(length(b),nA+i)=1;
        index_n(nA+i,1)=length(b);
    end
end
%% Úprava cen původních hran
j=0;
for i=1:nA % nA- matice je mA x nA
    j=j+1;
    index_nn(j,:)=index_n(i,:);
    nn(j)=i;
    ccn(j,1)=0;
end
%% Úprava cen a kapacit umělých hran
j=0;
for i=nA+1:(nA+length(b)-1)
    j=j+1;
    index_nb(j,:)=index_n(i,:);
    nb(j)=i;
    u(i)=abs(b(j));
    ccb(j,1)=1;
end
```

Pro účely přehledného popsání každé iterace v této úloze (viz odrázek 4.1), přiřadím každé hraně číslo na základě toho, kde se v matici **A** nachází. Využijeme toho, že každý sloupec matice **A** představuje hranu sítě. Hraně, jejíž sloupec je prvním sloupcem matice **A**, přiřadím číslo 1, hraně, jejíž sloupec je druhým sloupcem matice **A**, přiřadím číslo 2. Takto přiřadím každé hraně číslo (viz odrázek 4.4). Pro další obrázky bude platit následující značení (viz například obrázek 4.5): $\mathbf{n}_B$ - hrany, které máme v bázi, $\mathbf{n}_N$ - hrany, které uvažujeme pro vstup do báze, **plnacesta** - hrany, které jsme naplnili do maxima, **C** - vektor vyhodnocení směru hran v cyklu. V obrázku (viz například 4.6) a následujících obrázcích této kapitoly budou zvýrazněny některé hrany. Konkrétně zelená barva bude značit hranu, jejíž hodnota ve vektoru **d** je nejmenší (záporná), tedy bude vstupovat do báze (ještě v bázi není) a modrá barva bude značit hranu, která po vyhodnocení toků v cyklu (viz 4.1. - krok 2, cyklus vznikl přidáním hrany s minimální hodnotou ve vektoru **d** - zelená hrana) bázi opustí. Na obrázku 4.5 je vidět první přípustné řešení (s umělými hranami). Ještě zmiňme, že hodnoty u uzlů znamenají: hodnota uzlu; multiplikátor y_i, a hodnoty u hran: aktuální tok/max. tok; cena. Na obrázku 4.5 lze vidět, že nimimální hodnotu (v případě stejných hodnot můžeme vzít kteroukoli z nich, zde bereme vždy tu dřívější) má hrana 1. Tedy bereme 1 a hledáme cyklus, který vznikne jejím přidáním ke kostře grafu. Tento cyklus nám zde ukazuje vektor **C**, kladnost respektive zápornost hodnot hran určujeme tak, že projdeme cyklus ve směru hrany, kterou přidáváme, a pokud jdeme po směru bázové hrany, má kladnou hodnotu, pokud proti jejímu směru, má zápornou hodnotu. Je-li hodnota hrany kladná, pak je cyklus tvořen ve směru této



Obrázek 4.4: Uzly-červená, původní hrany-černá, hrany kvůli vyvážení-oranžová, umělé hrany-modrá.



$\mathbf{n}_B = (17, 18, 19, 20, 21, 22, 23)$
$\mathbf{n}_N = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (-2, -2, -2, -2, -2, -2, 0,0,0,0,0,0,0,-1,-1,-1)$
$\mathbf{plnacesta} = ()$
$\mathbf{C} = (1, -17, -19)$

Obrázek 4.5: Fáze 1, iterace 1.

hrany, a zjišťujeme tedy (a následně porovnáváme), jak velký tok lze do této hrany přidat než dosáhneme maxima $(u_{i,j} - x_{i,j})$. Je-li hodnota záporná, pak zjišťujeme (a následně porovnáváme) jak velký tok touto hranou teče (protože když se zvedá tok v naší vybrané nebázové hraně, tak v této hraně tok klesá). Tedy pro tento prípad máme hodnotu 15 pro uzel 17 a hodnotu 0 pro uzel 19. Tedy jak můžeme vidět na obrázku 4.6, do báze vejde hrana 1 a odejde hrana 19 a tuto hranu již dále neuvažujeme (umělá hrana).
Zjištění cyklu a vyhodnocení vektoru **C** v Matlabu:

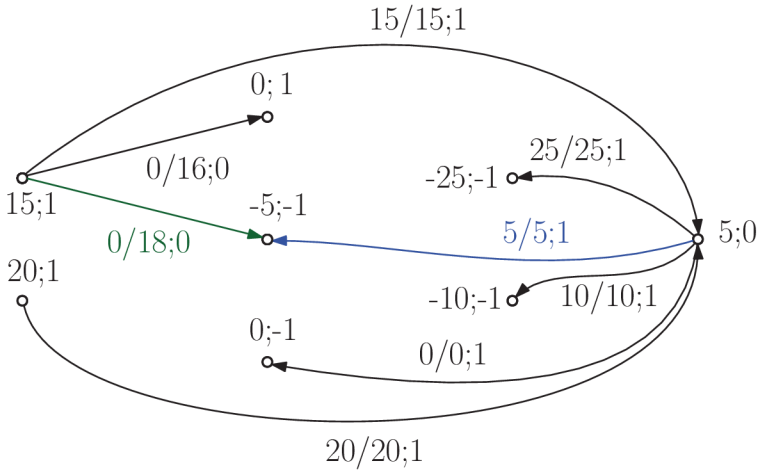
```
bb=b;
% tím že přičetu a odečetu 1 u uzlů imaginárně vytvořím hranu která má
vstoupit a nastavil její tok na 1
bb(index_nn(I,1))=bb(index_nn(I,1))-1;
bb(index_nn(I,2))=bb(index_nn(I,2))+1;
% úprava matice B na dolní trojuhelníkovou matici
```

```
[B_xx,b_xx,xx_posun]=Trojuhelnik_x(B,bb);
% výpočet toku
[xx_xx]=Vypocet_x(B_xx,b_xx,nB);
% při úpravě na trouhelnik jsem přehazoval proměnné takže teď je vracím
na místo kam patří
[xx]=Posunovani(xx_posun,xx_xx);
% odečtením nového toku (s imaginární hranou) od původního toku
identifikuji cyklus a směr hran
xx=xb-xx;
% vytvořím vektor C
C(1)=nn(I);
k=1;
for i=1:length(xx)
    if xx(i)>0
        k=k+1;
        C(k)=-nb(i);
    elseif xx(i)<0
        k=k+1;
        C(k)=nb(i);
    end
end
```

Vyhodnocení maximálního toku, který lze poslat hranou jež budeme přidávat:

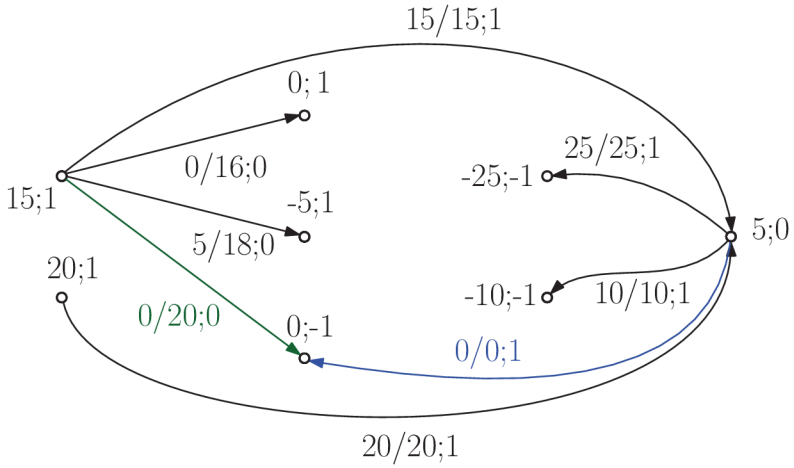
```
h=0; hh=0;
for i=1:length(C)
% hrana kterou přidávám
    if i==1
        h=h+1;
        u_minus_x(h)=(u(nn(I))-0);
% hrana vekteré se zvýší tok pokud hranou kterou chceme přidat poteče tok
    elseif C(i)>0
        h=h+1;
        poradi=find(nb==abs(C(i)));
        u_minus_x(h)=u(nb(poradi))-xb(poradi);
% hrana vekteré klesne tok pokud hranou kterou chceme přidat poteče tok
    elseif C(i)<0
        hh=hh+1;
        poradi=find(nb==abs(C(i)));
        jen_x(hh)=xb(poradi);
    end
end
% zjišťuji maximum z hran ve kterých bude přidáním vybrané hrany tok růst
(mají ve vektoru C kladnou hodnotu)
[M1,I1]=min(u_minus_x);
% zjišťuji maximum z hran ve kterých bude přidáním vybrané hrany tok kle-
sat (mají ve vektoru C zápornou hodnotu)
[M2,I2]=min(jen_x);
```

```
% zjišťuji maximální tok který budu schopen přidanou vybranou hranou
poslat
[M3,I3]=min([M2 M1]);
```



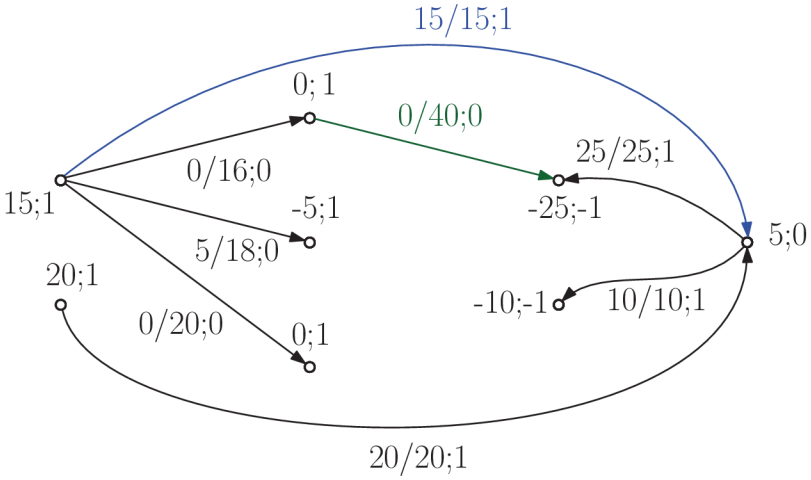
$\mathbf{n}_B = (1, 17, 18, 20, 21, 22, 23)$
$\mathbf{n}_N = (2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (-2, -2, 0, -2, -2, -2, -2, -2, 0, 0, 0, 0, -1, -1, -1)$
$\mathbf{plnacesta} = ()$
$\mathbf{C} = (2, -17, -20)$

Obrázek 4.6: Fáze 1, iterace 2.



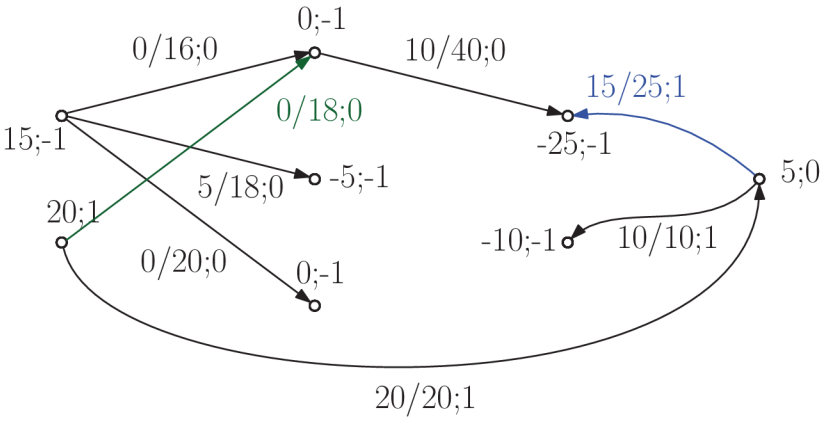
$\mathbf{n}_B = (1, 2, 17, 18, 21, 22, 23)$
$\mathbf{n}_N = (3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (-2, 0, 0, -2, 0, -2, -2, -2, -2, -2, 0, 1, -1, -1)$
$\mathbf{plnacesta} = ()$
$\mathbf{C} = (3, -17, -21)$

Obrázek 4.7: Fáze 1, iterace 3.



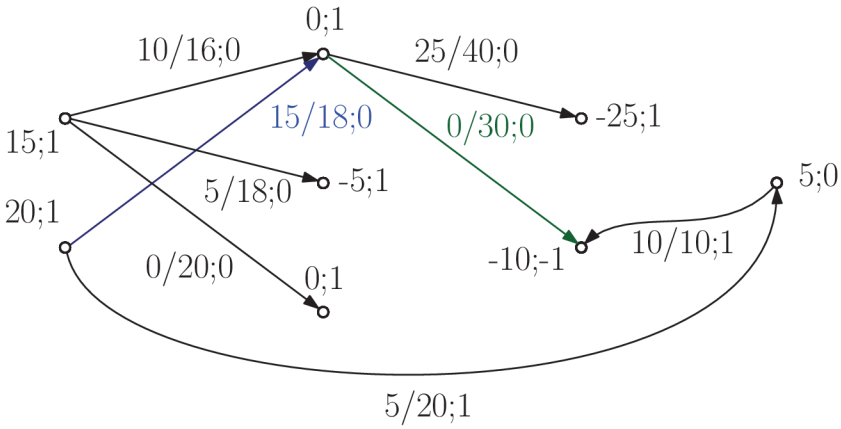
$\mathbf{n}_B = (1, 2, 3, 17, 18, 22, 23)$
$\mathbf{n}_N = (4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (0, 0, 0, 0, -2, -2, 0, -2, -2, -2, 1, -1, -1)$
plnacesta = ()
$\mathbf{C} = (8, 1, -17, -22)$

Obrázek 4.8: Fáze 1, iterace 4.



$\mathbf{n}_B = (1, 2, 3, 8, 18, 22, 23)$
$\mathbf{n}_N = (4, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (-2, -2, -2, 0, 0, 0, 0, 0, 0, -1, -1, -1)$
plnacesta = ()
$\mathbf{C} = (4, 8, -18, -22)$

Obrázek 4.9: Fáze 1, iterace 5.

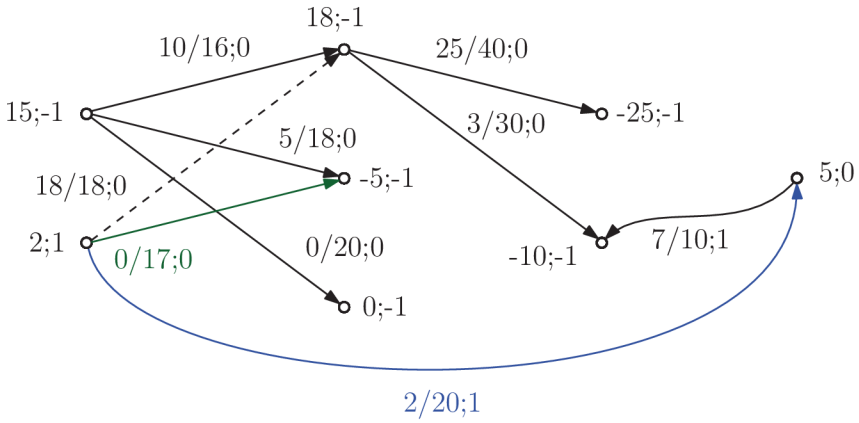


$\mathbf{n}_B = (1, 2, 3, 4, 8, 18, 23)$
$\mathbf{n}_N = (5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (0, 0, 0, -2, 0, 0, -2, -2, 1, 1, -1)$
plnacesta = ()
$\mathbf{C} = (9, 4, -18, -23)$

Obrázek 4.10: Fáze 1, iterace 6.

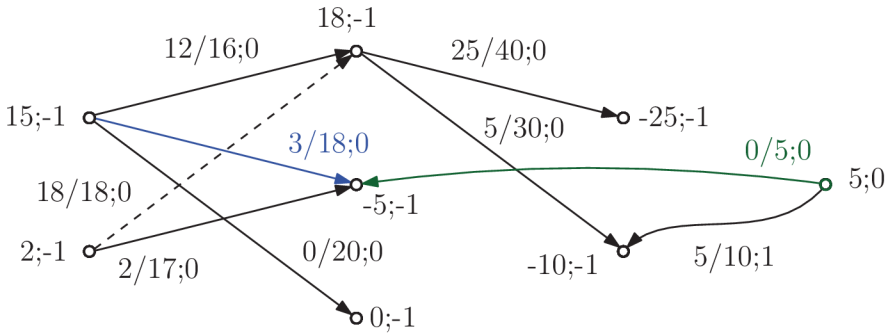
V iteraci 6 (viz obrázek 4.10) došlo k naplnění hrany do její plné kapacity. Již víme, že každé hraně bylo přiřazeno číslo (viz strana 23). Připomeňme si význam vektoru $\mathbf{C}$. První číslo vektoru je číslo hrany, která vstupuje do báze, ostatní jsou čísla hran tvořící cyklus. Kladnost (zápornost) čísel závisí na jejich orientaci vůči směru procházení cyklu.

Cyklus procházíme ve směru vstupující hrany (zelená). Pokud je orientace ostatních hran v cyklu ve směru, kterým cyklus procházíme, je jejich číslo kladné, pokud je jejich orientace opačná, je jejich číslo záporné. Pokud bychom zvyšovali tok zelenou hranou (číslo 9 ve vektoru **C**), bude se zvyšovat tok modrou hranou (číslo 4 ve vektoru **C**) a zároveň snižovat tok hranami číslo 18 a 23 (ve vektoru **C** -18 a -23). Zjišťujeme, že kapacita hrany 9 je $u_{3,7} = 30$, hrana 4 může zvýšit svůj tok o 3 $(18 - 15)$, hrany 18 a 23 jej mohou snížit o 5 (aktuální tok je roven 5) a 10 (aktuální tok je roven 10). Tedy minimální tok, který můžeme poslat hranou 9 (zelená) je 3. Zelená hrana vstoupí do báze s tokem $x_{3,7} = 3$, modrá hrana se naplní do maxima a naplněná odchází z báze. Pro další výpočet ji již neuvažujeme. To, že hrana směřující z uzlu 2 do uzlu 3 odchází z báze naplněná do maxima, demonstruji odečtením jejího toku ($x_{2,3} = 18$) od uzlu 2 a následným přičtením k uzlu 3 (viz 4.11). Přerušovaná hrana na obrázku 4.11 je tato dále neuvažovaná, do maxima naplněná hrana.



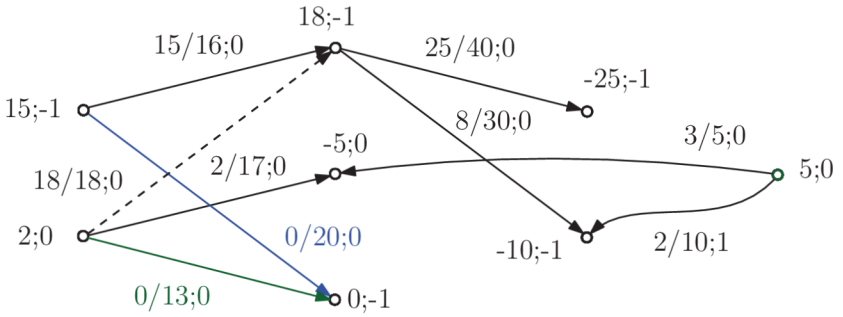
$\mathbf{n}_B = (1, 2, 3, 8, 9, 18, 23)$
$\mathbf{n}_N = (5, 6, 7, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (-2, -2, 0, 0, 0, 0, 0, -1, -1, -1)$
plnacesta $= (4)$
$\mathbf{C} = (5, 1, -2, 9, -18, -23)$

Obrázek 4.11: Fáze 1, iterace 7.



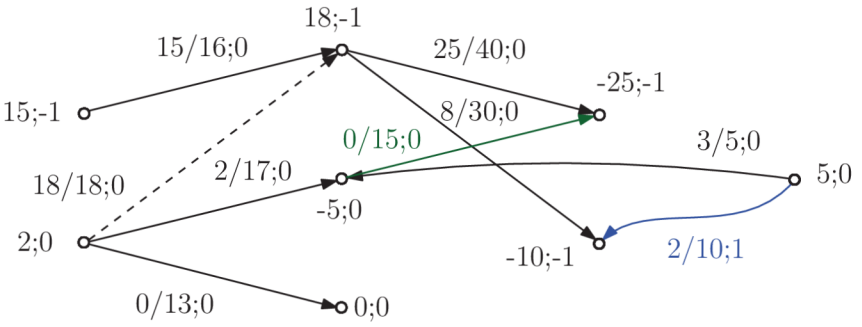
$\mathbf{n}_B = (1, 2, 3, 5, 8, 9, 23)$
$\mathbf{n}_N = (6, 7, 10, 11, 12, 13, 14, 15, 16)$
$\mathbf{d} = (0, 0, 0, 0, 0, 0, -1, -1, -1)$
plnacesta $= (4)$
$\mathbf{C} = (14, 1, -2, 9, -23)$

Obrázek 4.12: Fáze 1, iterace 8.



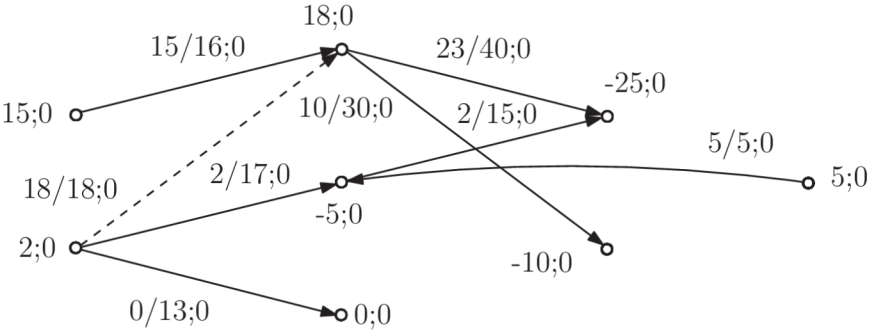
$\mathbf{n}_B = (1, 3, 5, 8, 9, 14, 23)$
$\mathbf{n}_N = (2, 6, 7, 10, 11, 12, 13, 15, 16)$
$\mathbf{d} = (1, -1, 1, -1, -1, -1, 0, -1, -1)$
plnacesta $= (4)$
$\mathbf{C} = (6, 1, -3, -5, 9, 14, -23)$

Obrázek 4.13: Fáze 1, iterace 9.



$\mathbf{n}_B = (1, 5, 6, 8, 9, 14, 23)$
$\mathbf{n}_N = (2, 3, 7, 10, 11, 12, 13, 15, 16)$
$\mathbf{d} = (1, 1, 1, 0, -1, -1, -1, -1, -1)$
plnacesta $= (4)$
$\mathbf{C} = (11, -8, 9, 14, -23)$

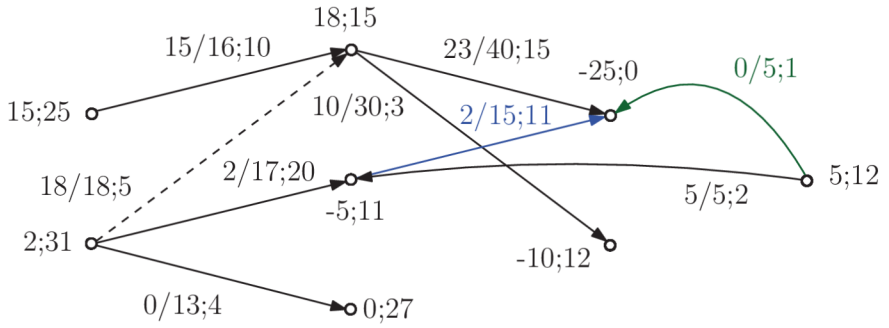
Obrázek 4.14: Fáze 1, iterace 10.



$\mathbf{n}_B = (1, 5, 6, 8, 9, 11, 14)$
$\mathbf{n}_N = (2, 3, 7, 10, 12, 13, 15, 16)$
$\mathbf{d} = (0, 0, 0, 0, 0, 0, 0, 0)$
plnacesta $= (4)$

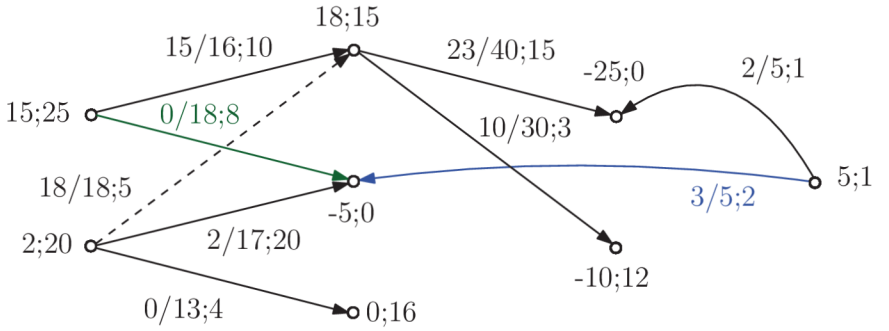
Obrázek 4.15: Fáze 1, iterace 11.

Na obrázku 4.15 jsme došli do optima ve smyslu maximalizace toku (vektor **d** obsahuje pouze nuly). Nyní tedy přejdeme k fázi 2. Vyhodnocovací procesy jsou stejné jako ve fázi 1 (viz 4.1.) s tím rozdílem, že nyní pracujeme s původními cenami hran.



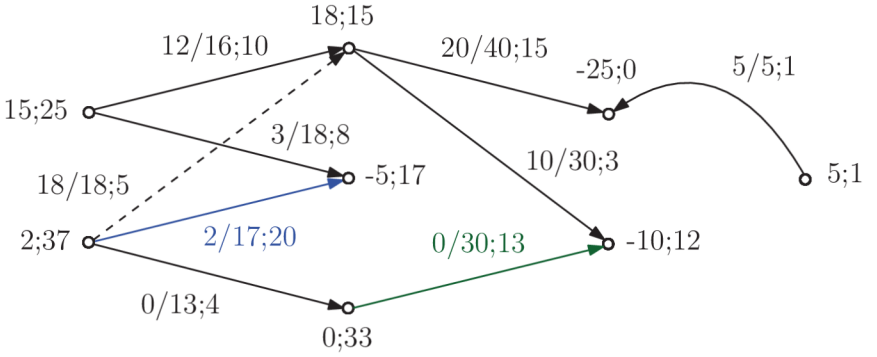
$\mathbf{n}_B = (1, 5, 6, 8, 9, 11, 14)$
$\mathbf{n}_N = (2, 3, 7, 10, 12, 13, 15, 16)$
$\mathbf{d} = (-6, 18, 5, 22, 9, -2, -11, 1)$
plnacesta $= (4)$
$\mathbf{C} = (15, -11, -14)$
$z = 682$

Obrázek 4.16: Fáze 2, iterace 1.



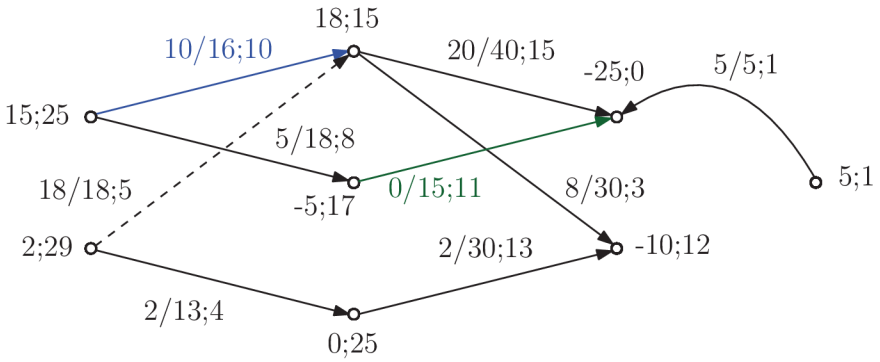
$\mathbf{n}_B = (1, 5, 6, 8, 9, 14, 15)$
$\mathbf{n}_N = (2, 3, 7, 10, 11, 12, 13, 14, 16)$
$\mathbf{d} = (-17, 7, -6, 22, 11, 20, 9, 12)$
plnacesta $= (4)$
$\mathbf{C} = (2, -1, -8, -14, 15)$
$z = 660$

Obrázek 4.17: Fáze 2, iterace 2.



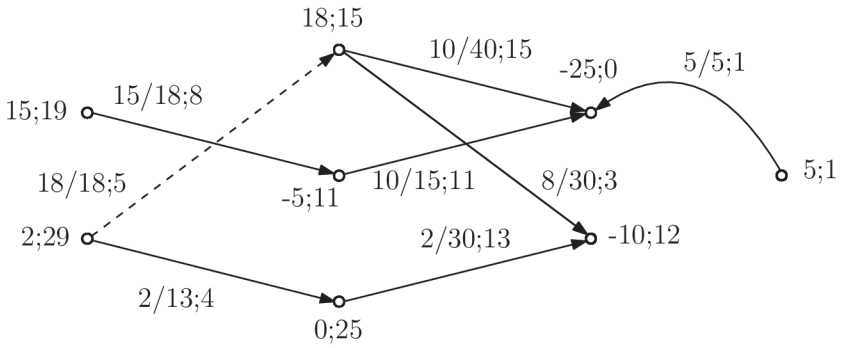
$\mathbf{n}_B = (1, 2, 5, 6, 8, 9, 15)$
$\mathbf{n}_N = (3, 7, 10, 11, 12, 13, 14, 16)$
$\mathbf{d} = (24, 11, 22, -6, 3, -8, 17, 12)$
plnacesta $= (4)$
$\mathbf{C} = (13, -1, 2, -5, 6, -9)$
$z = 609$

Obrázek 4.18: Fáze 2, iterace 3.



$\mathbf{n}_B = (1, 2, 6, 8, 9, 13, 15)$
$\mathbf{n}_N = (3, 5, 7, 10, 11, 12, 14, 16)$
$\mathbf{d} = (16, 8, 11, 14, -6, 3, 17, 12)$
$\mathbf{plnacesta} = (4)$
$\mathbf{C} = (11, -1, 2, -8)$
$z = 593$

Obrázek 4.19: Fáze 2, iterace 4.



$\mathbf{n}_B = (2, 6, 8, 9, 11, 13, 15)$
$\mathbf{n}_N = (1, 3, 5, 7, 10, 12, 14, 16)$
$\mathbf{d} = (6, 22, 2, 5, 20, 9, 11, 12)$
$\mathbf{plnacesta} = (4)$
$z = 533$

Obrázek 4.20: Fáze 2, iterace 5.

Na obrázku 4.20 si všimněme, že vektor $\mathbf{d}$ má všechny hodnoty nezáporné, tudíž bylo nalezeno optimální řešení této úlohy. Tok hranami, které na obrázku 4.20 nejsou zviditelněné, je nulový. Jak jsme mohli vidět u jednotlivých kroků, S.S.M. snižuje hodnotu účelové funkce v každém kroku.

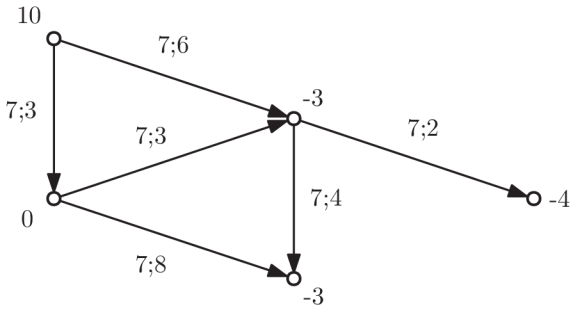
5. Srovnávání metod

Jak již bylo řečeno v kapitole 4, síťová simplexová metoda je vlastně speciálně upravená simplexová metoda pro síťové úlohy. Demonstrujme si tedy na několika příkladech užitečnost této úpravy. V této kapitole vždy uvedeme síť, která je řešena, a následně její tokové řešení, počet iterací a hodnoty minimalizované funkce, a to pro obě metody. Než se přesuneme k testovacím úlohám, uvedeme skript pro spouštění simplexové metody implementované v programu Matlab:

```
% nastavuji spodní mez hran (lb) na 0, pokud by nebyla nastavena (místo
lb by v zadávání bylo []), byla by defaultně nastavena na -inf
lb=zeros(1,length(c));
% nastavuji aby byl pro výpočet použit simplexový algoritmus
options = optimoptions('linprog','Algorithm','simplex')
% spouštím výpočet
[x,fval,exitflag,output,lambda] = linprog(c,A,b,[],[],lb,u,[],options)
```

5.1. Testovací úlohy

Jak je vidět na obrázku 5.1, tato úloha je velice jednoduchá. Hodnoty u uzlů představují vektor **b**, hodnoty u hran (g;h) pak jejich ohodnocení (g je kapacita hrany a h je její cena). Již zde je vidět efektivita S.S.M. (síťové simplexové metody), viz tabulka 5.1 (S.S.M. byla efektivnější o 1 iteraci).

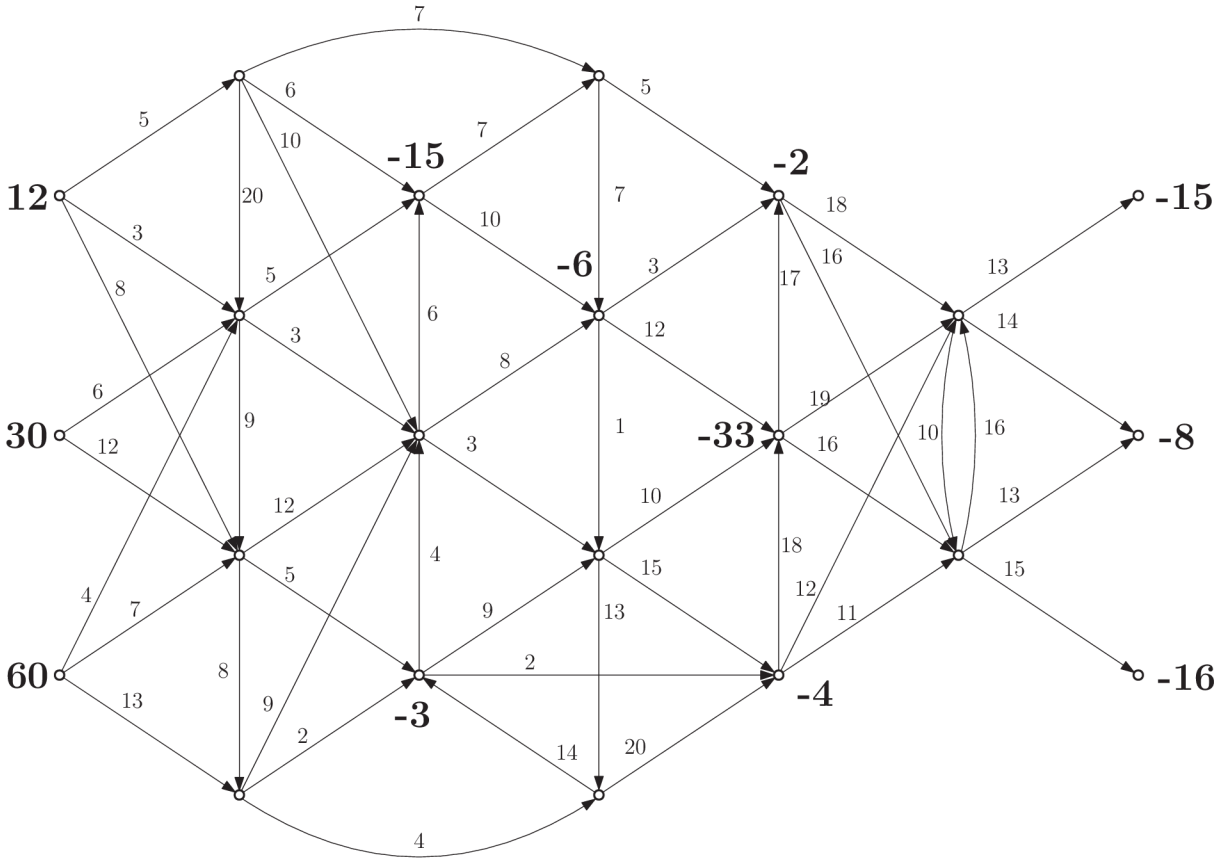


Obrázek 5.1: Velmi jednoduchá síť.

	S.M.	**S.S.M.**
Počet iterací	3	2
Řešení	$\mathbf{x} = (3, 7, 3, 0, 3, 4)$	
Hodnota funkce	$z = 80$	

Tabulka 5.1: Řešení sítě z obrázku 5.1.

Na obrázku 5.2 se nachází síť s 22 uzly a 53 hranami. Tučné hodnoty jsou hodnoty pro uzly (pro přehlednost nejsou napsány nulové hodnoty u uzlů - u kterého uzlu chybí hodnota, tam se nachází nula). Hrany u této sítě nejsou omezeny kapacitou, tudíž hodnoty u hran značí jejich cenu. V tabulce 5.2 lze vidět větší efektivitu (rozdíl počtu iterací byl 4) S.S.M. oproti S.M. než u sítě z obrázku 5.1.



Obrázek 5.2: Řešená síť.

	S.M.	S.S.M
Počet iterací	14	10
Čísla hran s nenulovým tokem	$2, 4, 6, 7, 14, 15, 18, 29,$ $30, 31, 32, 36, 46, 47, 49, 52, 53$	
Řešení pro hrany s nenulovým tokem	$\mathbf{x} = (12, 30, 14, 46, 15, 41, 46, 43,$ $39, 2, 33, 33, 15, 24, 15, 8, 16)$	
Hodnota funkce	$z = 2737$	

Tabulka 5.2: Řešení sítě z obrázku 5.2.

5.2. Reálná data

Pomocí dat poskytnudých Ústavem procesního inženýrství (tyto data poskytl program na základě vybraných krajů, pro které jsme chtěli vytvořit síť) jsem byl schopen vytvořit réálné sítě. Ilustrační příklad, jak vypadala vygenerovaná síť (v programu Microsoft Excel), je na obrázku 5.3. Z takto vygenerované sítě byla vzata data potřebná pro vytvoření vstupních údajů pro vytvořený program (viz obrázek 5.4).

	A	B	C	D	E	F	G	H	I
1	uzly	hrany	hodnota		hrany	vzdialenost		hrany	limit
2	10002	10002-10008	-1		10002-10008	45		10002-10008	100000
3	10008	10002-10008	1		10002-10009	27		10002-10009	100000
4	10002	10002-10009	-1		10002-10012	23		10002-10012	100000
5	10009	10002-10009	1		10003-10004	27		10003-10004	100000
6	10002	10002-10012	-1		10003-10007	31		10003-10007	100000
7	10012	10002-10012	1		10003-10010	46		10003-10010	100000
8	10003	10003-10004	-1		10003-10011	40		10003-10011	100000
9	10004	10003-10004	1		10003-10014	21		10003-10014	100000
10	10003	10003-10007	-1		10003-10015	26		10003-10015	100000

Obrázek 5.3: Ilustrační příklad vygenerované sítě.

	A	B	C	D	E	F	G	H	I	J
1	uzly		hodnota		uzly-uprava	hodnoty		cena		b
2	10001	10001-10002	-1		001	-1,00		102		13839,00
3	10002	10001-10002	1		002	1,00		152		14520,00
4	10001	10001-10005	-1		001	-1,00		97		-104659,00
5	10005	10001-10005	1		005	1,00		113		20423,00
6	10001	10001-10008	-1		001	-1,00		118		3847,00
7	10008	10001-10008	1		008	1,00		91		15019,00
8	10001	10001-10009	-1		001	-1,00		204		8869,00
9	10009	10001-10009	1		009	1,00		194		7305,00
10	10001	10001-10012	-1		001	-1,00		216		6132,00

Obrázek 5.4: Ilustrace převzatých dat.

Z tohoto souboru (viz obrázek 5.4) byla data následně načtena vytvořeným skriptem v programu Matlab:

```
% načtení matice AA
 filename = 'uloha.xlsx'; sheet = 1; xlRange = 'D2:E3797';
 AA = xlsread(filename,sheet,xlRange);

% vytvoření matice síte (matice A)
[m,n]=size(AA); k=0;
for i=1:2:m-1
    k=k+1; A(AA(i,1),k)=AA(i,2); A(AA(i+1,1),k)=AA(i+1,2);
end

% vytvoření vektoru c
  filename = 'Sešit1.xlsx'; sheet = 1; xlRange = 'G2:G1899';
  c = xlsread(filename,sheet,xlRange);


% vytvoření vektoru u
for i=1:length(c)
```

```
    u(i)=inf;
end

% vytvoreni vektoru b
filename = 'Sešit1.xlsx'; sheet = 1; xlRange = 'I2:I99';
b = xlsread(filename,sheet,xlRange);

% uprava matice A (při vytváření matice A mohli vzniknout prázdné řádky
- mažu je)
l=1;
while l~=0
    l=0; [mm,nn]=size(A);
    for i=1:mm-l
        if norm(A(i,:),1)==0
            l=l+1; A(i,:)=[];
        end
    end
end
```

V prvním případě byla vytvořena síť z krajů, které tvořily souvislou oblast, konkrétně to byly: Jihomoravský, Moravskoslezský, Olomoucký a Zlínský. Takto vytvořená síť byla složena z 69 uzlů a 456 hran. Vytvořená vstupní data programu lze nalézt v příloze 1. V tabulce 5.3 je vidět minimalizovaná hodnota z a porovnání zkoumaných metod pro tuto síť (S.S.M byla efektivnější o 11 iterací). Vypočtený tok je vektor o 456 složkách, byl tedy také umístěm do přílohy 1.

	S.M.	**S.S.M**
Počet iterací	66	55
Hodnota funkce	$z = 19207261$	

Tabulka 5.3: Řešení sítě pro první vytvořenou síť.

Ve druhém případě byla vytvořena síť pro náhodně vybrané kraje, konkrétně to byly: Jihočeský, Jihomoravský, Karlovarský, Královehradecký, Liberecký, Moravskoslezský a Praha. Takto vytvořená síť byla složena z 93 uzlů a 960 hran. Vytvořená vstupní data programu lze nalézt v příloze 2. V tabulce 5.4 je vidět minimalizovaná hodnota z a porovnání zkoumaných metod pro tuto síť (S.S.M byla efektivnější o 5 iterací). Stejně jako v předchozím případě je vypočtený tok velký vektor (960 složek), lze ho nalézt v příloze 2.

	S.M.	**S.S.M**
Počet iterací	74	68
Hodnota funkce	$z = 80365308$	

Tabulka 5.4: Řešení sítě pro druhou vytvořenou síť.

Jak je vidět v tabulkách 5.1, 5.2, 5.3 a 5.4 síťová simplexová metoda potvrdila předpoklad větší efektivnosti oproti simplexové metodě (S.S.M je modifikace S.M. pro úlohy na síti). U testovacích úloh (viz v tabulky 5.1 a 5.2) si lze povšimnout, že minimalizovaná hodnota z i vypočtený tok získané S.S.M. a S.M. se pro každou úlohu shodují. U úloh

s reálnými daty (viz tabulky 5.3 a 5.4 a příslušné přílohy) si lze povšimnout , že minimalizovaná hodnota z je pro obě metody stejná, ale vypočtený tok S.S.M. a S.M se liší.

6. Závěr

Byly shrnuty poznatky z teorie grafů a optimalizace a následně uvedeny myšlenky řešení S.M. a S.S.M na příkladech. Pro každou z metod byla část práce věnována řešení a ilustraci těchto myšlenek na konkrétním příkladu.

Velká část úsilí vynaloženého na vypracovávání této práce byla zaměřena na vytvoření funkčního programu řešícího síťovou simplexovou metodu a na vytvoření pomocných skriptů (skript pro nastavení a spuštění funkce *linprog*, skript pro úpravu vygenerovaných dat na vstupy programu řešícího síťovou simplexovou metodu) a jejich úpravám a následné optimalizaci.

U řešení testovacích úloh i úloh na reálné síti je ilustrována větší efektivita (nižší počet iterací) S.S.M. oproti S.M. (pro výpočet byla použita funkce *linprog* implementovaná v programu Matlab) za stejné hodnoty minimalizované účelové funkce z. U testovacích úloh byl vypočtený tok sítí stejný pro obě metody, u úloh na reálné síti se lišil.

Literatura

[1] BAZARAA, M. S., John J. JARVIS a Hanif D. SHERALI. *Linear programming and network flows. 4th ed.* Hoboken, N.J.: John Wiley, c2010. ISBN 978-0-470-46272-0.

[2] GHIANI, Gianpaolo., Gilbert LAPORTE a Roberto. MUSMANNO. *Introduction to logistics systems planning and control.* Hoboken, N.J, USA: John Wiley, c2004. ISBN 0-470-84917-7.

[3] GRIVA, Igor, STEPHEN G. NASH a Ariela SOFER.*Linear and nonlinear optimization. 2nd ed.* Philadelphia, Pa: Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 2009. ISBN 0898717736.

[4] KOVÁŘ, P.:*Úvod do Teorie grafů.*[Učební text.] Vysoká škola báňská – Technická univerzita Ostrava, 2016. 167s.

[5] SKLENÁŘ, J.:*Network Flow Models and Integer Programming.* [Sylabus.] University of Malta, Faculty of Science, 2004. 21s.

[6] WILLIAMS, H. P.*Model building in mathematical programming. 5th ed.* Hoboken, N.J.: Wiley, 2013. ISBN 9781118443330.

7. Seznam příloh

Příloha 1 - Vstupní data a vypočtený tok pro první úlohu kapitoly 5.2.
Příloha 2 - Vstupní data a vypočtený tok pro druhou úlohu kapitoly 5.2.
Příloha 3 - Vytvořený program řešící S.S.M a pomocné skripty