Algovize

Luděk Kučera

12. ledna 2008

1 Simplexový algoritmus

V úvodu této kapitoly je nutno uvést, že tento applet ukazuje pouze jednu partii, která je v teorii lineárních programů důležitá. Vůbec se například nebudu zabývat dualitou v lineárním programování, což je zdaleka nejdůležitější součást celé teorie, protože je to svojí podstatou matematický vztah, který neumím výstižně vizualizovat. Z tohoto důvodu také nebudou probírány další důležité pojmy primární a duální úlohy a z nich vyplývající metody výpočtu. Zcela mimo zůstanou i otázky celočíselnosti řešení a aplikace lineárního programování a také algoritmy pracující zaručeně v polynomiálním čase, jako je například elipsoidová metoda a metody vnitřního bodu (na ty ale možná časem dojde). Zaměřím se zde na výklad základních myšlenek simplexového algoritmu.

Při definici problému i popisu algoritmu se nebudu striktně držet obvyklé symboliky a formalismu, které jsou výhodné například při odvozování duality, ale zbytečně zatěžují čtenáře obraty, které nejsou důležité pro pochopení myšlenky simplexového algoritmu. Lineární programování budu chápat jako lineární optimalizaci nad obecným polyedrem a simplexový algoritmus jako putování mezi vrcholy polyedru, které je monotónní vzhledem k optimalizovanému funkcionálu.

V této kapitole se nejprve budeme zabývat optimalizací v rovině a později v třídimenzionálním prostoru, neboť tyto případy je možno snadno vizualizovat a vybudovat tak geometrický přístup k LP a simplexovému algoritmu konkrétně, který pak je snadné zobecnit na vícedimenzionální případ.

Scéna: *Přímky v rovině*

Body roviny, vyhovující rovnici $ax + by = c$, kde a a b nejsou současně rovna nule (a x a y představují x-ovou a y-ovou souřadnici uvažovaného bodu) vytvářejí přímku a naopak každá přímka se takto dá popsat.

V této scéně je znázorněna souřadná soustava roviny s osami x a y opatřenými měřítkem. Nakreslete přímku tak, že v jistém místě obrazovky stisknete myš a v jiném místě ji uvolníte. Algovize body spojí přímkou a připíše k ní její rovnici. Než bude knoflík myši uvolněn, je přímka kreslena červenou barvou, místo kde byla myš stisknuta je označeno malým kolečkem a okamžitá poloha kurzoru je označena šipkou. V okamžiku uvolnění myši kolečko a šipka zmizí a přímka je zafixována a kreslena tmavě modře. Kreslení přímky lze vícenásobně opakovat. Současně s nakreslením přímky se na straně obrazovky objeví pole, ve kterém je napsána její rovnice (pro právě kreslenou přímku žluté a pro nakreslené přímky bílé).

V kontrolním panelu je ale možno zvolit i další módy funkce myši. Kromě již popsané volby [**Kresli přímku**] lze také vybrat volbu [**Zvol a edituj přímku**], [**Vynech přímku**] a [**Posuň nerovnosti**].

Při volbě [**Zvol a edituj přímku**] se některá z nakreslených přímek zvolí tím, že se na ni klepne myší. Zvolená přímka ze zobrazí fialově s kolečkem a šipkou, jak na ní byly v posledním okamžiku, kdy byla kreslena. Současně se barevně odliší její rovnice (žluté pole). Někdy tuto funkci volíme jen proto, abychom si přiřadili zvolené přímce její rovnici; jindy ale použijeme možnosti přímku změnit tím, že tažením myší přemístíme její kolečko nebo šipku. Pokud klepneme jinam než na zvolenou přímku, volba přímky se anuluje (a bylo-li klepnuto na jinou přímku, tato je zvolena a může vzápětí být editována).

Při volbě [**Vynech přímku**] se klepnutím na přímku tato přímka vynechá. Při této volbě se také objeví jinak skrytý knoflík [**Vynech vše**], kterým je možno vynechat všechny nakreslené přímky a popřípadě (po změně volby) znovu začít přímky kreslit.

Při volbě [**Posuň nerovnosti**] je možno myší přetáhnout blok nerovností do jiné polohy, nic jiného se myší nedá dělat. Knoflíky [**+**] a [**-**] se dá velikost okénka s nerovnostmi měnit a vypnutím [**Ukaž nerovnosti**] je možné zobrazování nerovností vypnout. Zřejmým způsobem lze i skrýt osy souřadnic.

Poznamenejme, že tutéž přímku lze popsat mnoha rovnicemi. Jestliže všechny tři parametry a, b a c v rovnici přímky vynásobíme týmž nenulovým číslem, dostaneme jiný popis téže přímky (a naopak lze od jednoho popisu k druhému tímto způsobem vždy přejít). Popis uváděný na obrazovce je volen tak, aby platilo $|a| + |b| = 1000$, čímž je rovnice přímky jednoznačně určena.

Scéna: *Poloroviny*

Přímka rozetne rovinu na dvě části - poloroviny. Polorovinu chápeme tak, že zahrnuje i hraniční přímku (je to tedy *uzavřená polorovina*). Průsečíkem obou polorovin je tedy přímka, která je určuje.

Poloroviny určené přímkou s rovnicí $ax+by = c$ je možno popsat jako množiny bodů $[x, y]$, které vyhovují nerovnosti $ax + by \leq c$ respektive $ax + by \geq c$. Druhou nerovnost je možné psát také jako $(-a)x + (-b)y \leq -c$ a proto vhodnou volbou kladných a záporných koeficientů je možno popsat každou polorovinu nerovností se znaménkem $\leq$ a konstantou na pravé straně a tím zjednodušit a sjednotit formalismus zápisu.

Nakreslete stejným způsobem jako v předchozí scéně přímku. Rozdělení na poloroviny se barevně zvýrazní. (Zopakujme, že každá z polorovin zahrnuje i hraniční přímku, jak je z rovnic ostatně vidět). V dalším budeme vždy uvažovat polorovinu znázorněnou zeleně. Místo šipky v této scéně používáme pro znázornění orientace přímky pološipku, která leží celá v uvažované zelené polorovině.

Pokud není knoflík myši stisknut, v zobrazení rovnice je uvedena i hodnota levé strany pro bod, daný okamžitou polohou kurzoru. Barevně je také rozlišeno, zda nerovnost platí. Je-li splněna jako ostrá nerovnost, je okénko nerovnosti světle zelené, platí-li rovnost, je tmavě zelené. Pokud nerovnost neplatí, je okénko fialové. Ověřte si, že okénko je (světle nebo tmavě) zelené, pokud kurzor ukazuje do zvolené poloroviny a je fialové, je-li mimo ni.

Na rozdíl od předchozího případu se nakreslením nové přímky předchozí přímka i rozdělení roviny jí určené vymaže.

Scéna: *Lineární nerovnosti a konvexní mnohoúhelník*

V této kapitole se budeme zabývat konvexními mnohoúhelníky. Konvexní mnohoúhelník v rovině je množina bodů $[x, y]$, které vyhovují soustavě lineárních nerovností

$$a_1x + b_1y \leq c_1$$

$$a_2x + b_2y \leq c_2$$

$$\dots$$

$$a_mx + b_my \leq c_m$$

kde $a_1, b_1, c_1, \dots, a_m, b_m, c_m$ jsou konstanty. Jelikož množina bodů, které vyhovují jedné z těchto nerovností, je polorovina, je konvexní mnohoúhelník v rovině průnikem polorovin.

V této scéně je možno konvexní mnohoúhelník budovat postupným přidáváním lineárních nerovností. Přímka se kreslí stejně jako v předchozí scéně, ale nakreslením nové přímky dříve nakreslené přímky nezmizí, ale zůstávají na obrazovce. Je možno je editovat jako tomu bylo v první scéně.

Vidíme tedy současně mnoho zvolených polorovin. Body, které neleží přímo na přímkách jsou znázorněny takto: Body, které leží ve všech polorovinách a tedy představují konvexní mnohoúhelník, daný jejich průnikem, jsou zelené. Zelená oblast je tedy mnohoúhelník definovaný nerovnostmi v rámečku. Ostatní body jsou modré, ale je odlišeno, v kolika polorovinách leží. Body, které leží v hodně polorovinách (ale ne ve všech), jsou tmavě modré, tím tmavší, čím větší počet polorovin je obsahuje. Naopak body, ležící v málo polorovinách jsou světlé a ty, které nejsou v žádné (možná ale takové body nejsou) jsou takřka bílé. Pokud není stisknut knoflík myši, pak pro bod ukazovaný kurzorem jsou také barevně odlišeny nerovnosti stejným způsobem jako v minulé scéně. Pohybujte kurzorem a sledujte, které nerovnosti jsou splněny a které ne.

Knoflíkem [**Nový polyedr**] je také možno vytvořit náhodně nový polyedr, který je pak možno upravovat přidáváním, ubíráním nebo přesouváním hraničních přímek. Checkboxem [**Pouze polyedr**] se přepne do módu, kdy všechny body mimo zelený mnohoúhelník mají stejnou barvu. Není pak rozlišeno v kolika polorovinách jsou obsaženy, ale jen zda patří do všech nebo ne. Tato volba je užitečná pokud nás zajímá jen samotný mnohoúhelník polorovinami vytvořený.

Mnohoúhelník může být neomezený, omezený a přitom neprázdný a nebo prázdný. Pro jednu nerovnost ($m = 1$) je tvořen neomezenou polorovinou, pro dvě nerovnosti je obecně představován neomezeným klínem (ale pro rovnoběžné hraniční přímky může být tvořen polorovinou, nekonečným pásem konstantní šířky, přímkou nebo být prázdný). Zkuste nakreslit všechny tyto možnosti.

Počínaje třemi nerovnostmi mnohoúhelník může (ale nemusí) být omezený nebo i prázdný i v případě nerovnoběžných hraničních přímek.

Přidáním další nerovnosti k soustavě, která má alespoň jedno řešení (neprázdný mnohoúhelník) se stane jedna z následujících možností

1. mnohoúhelník se zmenší, ale zůstane neprázdný - to znamená, že některé body původního mnohoúhelníku nové nerovnosti vyhovují a některé ne,

2. mnohoúhelník se nezmění - všechny body mnohoúhelníku nové rovnici vyhovují, což znamená, že platnost nové nerovnosti lze odvodit z nerovností předcházejících,

3. nové nerovnosti nevyhovuje žádný bod stávajícího mnohoúhelníku - mnohoúhelník se stane prázdným, nová nerovnost je ve sporu s předcházejícími.

Scéna: *Lineární funkcionál*

Lineární funkcionál je funkce, která každému bodu $[x, y]$ roviny přiřadí číslo $Ax + By$, kde A a B jsou konstanty. Na displeji jsou graficky znázorněny hodnoty funkcionálu s náhodně volenými hodnotami A a B. Hodnoty jsou znázorněny pomocí barev tepelné škály: nejvyšší hodnoty jsou znázorněny bíle, přecházejí do stále temnější žluté, pak červené, z nejtmavší červené do tmavnoucí modré, až se nakonec stanou černými (ne všechny barvy musí být viditelné). Je také nakreslena souřadná soustava s měřítky a šipka představující vektor (A, B), který začíná v počátku a vede do bodu $[A, B]$.

Budete-li pohybovat kurzorem se stisknutým knoflíkem myši, konec vektoru bude vždy v místě kurzoru. Výjimkou je, pokud je kurzor příliš blízko počátku - bráním se zobrazování funkcionálu daného nulovým vektorem a nebo pokud je příliš daleko - barevné pole by bylo úzké a na okraji displeje by se nedostávaly barvy.

Je vidět, že hodnoty funkcionálu se nejrychleji mění ve směru vektoru (A, B), zatímco ve směru na něj kolmém se nemění. Je to dáno tím, že hodnota funkcionálu v bodě $[x, y]$ je dána skalárním součinem vektoru (A, B) s vektorem (x, y), začínajícím v počátku a končícím v bodě $[x, y]$. Skalární součin vektoru (A, B) s přírůstkem, který je na něj kolmý, je nulový, zatímco největší je pokud jsou oba vektory kolineární.

Na obrazovce jsou také znázorněny některé "vrstevnice", čáry spojující body mající stejnou hodnotu funkcionálu. Jsou tvořeny šedě nakreslenými přímkami, které jsou kolmé na vektor určující směr nejrychlejšího přírůstku funkcionálu a jsou tedy navzájem rovnoběžné.

Je zřejmé, že čím je vektor $[A, B]$ delší, tím rychleji funkcionál v jeho směru roste, naopak pro velmi krátký vektor je funkcionál v pozorovatelné oblasti skoro konstantní.

Jak uvidíte dále, pro naše účely nebude důležité, jak rychle funkcionál roste, ale ve kterém směru roste nejrychleji. Jinými slovy, bude nás zajímat pouze směr, ale nikoli délka vektoru (A, B), který určuje funkcionál.

V rohu obrazovky je malé okénko, které ukazuje hodnotu funkcionálu v místě ukazovaném kurzorem. Ještě jednou opakuji, že důležitá není absolutní hodnota funkcionálu, ale její změny při pohybu kurzoru. Se stisknutým knoflíkem myši zkuste měnit směr vektoru, určujícího funkcionál a pak s uvolněným knoflíkem sledujte změnu hodnoty funkcionálu při pohybu kurzorem.

Scéna: *Lineární optimalizace v rovině*

V této scéně se dostáváme k základní úloze, kterou budeme v rovině řešit.Na displeji je nyní zobrazen konvexní mnohoúhelník a současně lineární funkcionál a máme nalézt ten bod mnohoúhelníku, pro který je hodnota funkcionálu největší.

Mnohoúhelník na displeji je vygenerován náhodně a zobrazen zeleně. Jako podklad pod ním je provedeno grafické znázornění funkcionálu využitím tepelné škály a vrstevnic, jak byly vysvětleny v předchozí scéně.

Řešením problému lineární optimalizace je ten bod mnohoúhelníku, který leží na nejsvětlejším možném místě funkcionálu. Bod, který je řešením, je znázorněn malým červeným kolečkem. Pohybem kurzoru při stisknutém knoflíku myši měňte funkcionál a sledujte řešení.

Jistě je vám zřejmé, že řešením je vždy některý vrchol mnohoúhelníku, tedy místo, kde se hranice mnohoúhelníku zalamuje. Jen občas dojde k výjimce, kdy množinou řešení je celá jedna hrana mnohoúhelníku, spojující dva jeho sousedící vrcholy.

Pro změnu zadání stiskněte knoflíku [**Nový polyedr**], ukáže se jiný konvexní mnohoúhelník a funkcionál a opět je možno zobrazit si řešení. Volbou na panelu ovladače lze vybrat, zda má mnohoúhelník být omezený nebo neomezený. Pro zjednodušení scény není implementována možnost vytvářet nebo měnit mnohoúhelník kreslením, editací nebo vynecháváním hraničních přímek.

Jistě vám je jasné, že pokud je zelený konvexní mnohoúhelník neprázdný a omezený, má úloha lineární optimalizace vždy řešení; řešením je obvykle jediný vrchol mnohoúhelníka, méně často jeho hrana (jen se může stát, že řešení leží mimo obrazovku). Pokud ale je mnohoúhelník neomezený, řešení může existovat, ale také nemusí. Druhý případ nastává, pokud vektor určující funkcionál ukazuje směrem, který svírá ostrý úhel s alespoň jednou polopřímkou, která leží celá v neomezeném mnohoúhelníku, protože podle takové polopřímky rostou hodnoty funkcionálu do nekonečna, takže konečné maximum hodnot funkcionálu neexistuje.

Úloha nalézt řešení lineárního programu v rovině, tedy se dvěma proměnnými x a y, není složitá a i pro velký počet nerovností by nebylo těžké ji vyřešit i jinými způsoby, než je metoda popisovaná v dalších scénách, ale postup, který ukážeme, je snadné zobecnit pro libovolný počet proměnných a pod názvem "simplexový algoritmus" je i v současné době patrně nejpoužívanější metodou pro řešení lineárních programů i přes to, že byla popsána G. B. Danzingem v již roce 1947.

Scéna: *Pohyb po hraně v rovině*

Jak jsme viděli v předchozí scéně, pokud má úloha lineárního programování řešení, tak toto řešení leží na obvodu mnohoúhelníka daného nerovnostmi a je jím vrchol tohoto mnohoúhelníka nebo, pokud je řešení více, pak řešeními jsou dva sousedící vrcholy mnohoúhelníka a všechny body hrany mezi nimi. Simplexový algoritmus hledá řešení tak, že začne v libovolném vrcholu mnohoúhelníka daného nerovnostmi a pohybuje se po obvodu mnohoúhelníka, dokud se nedostane do vrcholu, který je řešením (respektive jedním z řešení).

Podívejme se nyní na to, jak probíhá přechod z jednoho vrcholu mnohoúhelníku do vrcholu sousedního. Poznamenávám, že v této scéně je mnohoúhelník vždy neprázdný a omezený. Na obrazovce se objevil žlutý kotouček, jehož střed bude stále na obvodu mnohoúhelníka. Kotoučkem je možno pohybovat tak, že klepnete na knoflík [**Jdi vlevo**] nebo [**Jdi vpravo**] a kotouček se dá do pohybu po obvodu mnohoúhelníka do sousedního vrcholu ve zvoleném směru. "Jít vpravo" znamená ve směru hodinových ručiček, protože tak je nutné točit volantem, chceme-li zatočit vpravo, podobně vlevo je proti směru hodinových ručiček. Chvíli si pohyb po obvodu zkoušejte.

Nebaví-li vás znázorněný mnohoúhelník, můžete si nechat zkonstruovat jiný knoflíkem [**Nový polyedr**]. Nejspíš vás ale nebaví tato scéna, vždyť je to tak jednoduché. Podíváme se ale ještě, jak určit souřadnice souseda daného vrcholu ve směru dané hrany, když se nedíváme na obrázek, ale chystáme se vysvětlovaný algoritmus programovat a máme k dispozici jen nerovnosti určující mnohoúhelník a souřadnice vrcholu s kotoučkem.

Místo knoflíků [**Jdi vlevo/vpravo**] stiskněte jeden z knoflíků [**Vysvětli vlevo**] nebo [**Vysvětli vpravo**]. Místo aby se kotouček přesunul do jednoho ze sousedních vrcholů se displeji nakreslí vše potřebné pro ilustraci následujícího výkladu, který si klade za cíl zjistit vrchol do kterého by se kotouček při stisknutí knoflíku přesunul:

- Na displeji se objevily všechny přímky, které procházejí hranami mnohoúhelníka. Jsou to přímky, jejich rovnice dostaneme tak, že v nerovnostech určujících mnohoúhelník nahradíme symbol $\leq$ symbolem $=$ (rovnost).
- Z vrcholu, ve kterém je kotouček, vycházejí dvě hrany. Ta po které by měl přejít kotouček po stisku knoflíku [**Jdi ...**] a která je nyní nakreslena jako červená šipka a je součástí přímky nakreslené červeně, se bude nazývat *pivotní*, druhá hrana je *nepivotní*.
- červenými a černými kolečky jsou označeny průsečíky červené přímky, procházející pivotní hranou, s tmavomodrými přímkami procházejícími ostatními hranami (s tím, že některé průsečíky leží mimo obrazovku).
- Jak jste si jistě všimli, polorovina určená nerovností, která odpovídá nepivotní hraně, se zbarvila tmavěji než je původní barva pozadí (ale je částečně zakryta mnohoúhelníkem). Průsečíky zmíněné v předchozím odstavci jsou kresleny červeně, pokud leží v tmavě modré polorovině dané nepivotní hranou, a jsou kresleny jako černé v opačném případě.
- Vrchol pivotní hrany, do kterého by se kotouček přesunul, je ten z bodů označených červeným kolečkem, který nejbližší vrcholu se žlutým kotoučkem.

Určení průsečíku přímek procházejících dvěma vybranými hranami je přitom jednoduché: hranám odpovídají nerovnosti ze souboru, určujícího mnohoúhelník. Nahradíme-li v nerovnostech $\leq$ symbolem $=$, dostaneme soustavu dvou lineárních rovnic o dvou neznámých a jejich vyřešením se získají souřadnice průsečíku.

Určení, které průsečíky leží v polorovině dané nerovností odpovídající nepivotní hraně, je ještě jednodušší: souřadnice průsečíku se dosadí do příslušné nerovnosti a zjistí se, zda je splněna.

Výklad právě uvedený můžete přeskočit, pokud Vám jde jen o pochopení myšlenky simplexového algoritmu. Jestliže však budete chtít algoritmus také programovat, dobře si výklad promyslete.

Scéna: *Pohyb po hraně neomezeného mnohoúhelníka*

Scéna je opakováním předchozí. Jistě ale vám je zřejmé, že některé hrany neomezeného mnohoúhelníka jsou neomezené polopřímky. Pustí-li se po takové hraně kotouček, nikdy nedorazí do sousedního vrcholu, ale navždy by běžel po polopřímce. Musíte jej proto dostat zpět stisknutím knoflíku pro opačný směr.

I zde je možno přepnou do módu [**Výklad**]; uvidíte, že pokud by kotouček měl běžet do nekonečna po hraně tvořené polopřímkou, pak nevznikne žádný červeně zvýrazněný průsečík, neboť polopřímka tvořící hranu (což je průsečík červené přímky a tmavé poloroviny určené nerovností odpovídající nepivotní hraně) se neprotíná z žádnou z přímek procházejících ostatními hranami.

Scéna: *Simplexový algoritmus v rovině*

Scéna je do jisté míry opakováním předchozích dvou - ukazuje konvexní mnohoúhelník a kotouček, který se pohybuje po jeho obvodu; zde jím ale pohybuje program. Navíc je zde ale zobrazen i lineární funkcionál a program kotoučkem pohybuje tak, aby vždy mířil k vyšším hodnotám funkcionálu.

Je-li na ovladači volba [**Funkcionál**], lze myší měnit lineární funkcionál, při volbě [**Polož kotouček**] lze nastavit výchozí polohu žlutého kotoučku klepnutím na některý vrchol mnohoúhelníka.

Po nastavení výchozí situace klepněte na knoflík [**Počítej**]. Algovize určí optimum funkcionálu na mnohoúhelníku pomocí simplexového algoritmu a přepne do animačního módu, kde lze výpočet krokovat nebo nechat volně běžet. Každý krok je představován přechodem kotoučku po hraně mnohoúhelníka a kroky se provádějí tak dlouho, dokud není nalezeno optimální řešení problému; pak se kotouček zastaví a zčervená. Knoflíkem [**Zruš**] se vrátíte do módu nastavení a můžete zvolit jinou výchozí situaci nebo přejít do další scény.

Jelikož optimalizujeme v rovině, z každého vrcholu mnohoúhelníka vycházejí dvě hrany. Nejčastěji v průběhu výpočtu kotouček leží ve vrcholu, ze kterého jedna hrana směřuje k vyšším hodnotám funkcionálu a druhá (pokud nejsme na začátku výpočtu, je to ta, po které jsme do vrcholu přišli), jde k nižším hodnotám. Pak je jasné, kam kotouček posouvat. Na začátku výpočtu někdy mohou oba směry stoupat k vyšším hodnotám funkcionálu, pak je v zásadě jedno, kudy půjdeme. Jestliže v obou směrech funkcionál klesá (nebo alespoň neroste), je kotouček v místě nejvyšší hodnoty funkcionálu, tedy bylo nalezeno optimum a výpočet končí.

Scéna: *Simplexový algoritmus - neomezený mnohoúhelník*

Předchozí scéna je opakována s tím, že nyní budou všechny vytvořené konvexní mnohoúhelníky neomezené. Pokud se stane, že lineární funkcionál stoupá podél nějaké neomezené polopřímky, pak může nabývat libovolně vysokých hodnot a bod mnohoúhelníka s maximální hodnotou funkcionálu neexistuje.

Scéna: *Simplexuj si sám v rovině*

Scéna je stejná jako předchozí dvě, ale kotoučkem nepohybuje program, ale uživatel pomocí knoflíků [**Jdi vpravo**] a [**Jdi vlevo**] tak, jak to bylo ukázáno v předminulé scéně. Je třeba kotoučkem pohybovat tak, aby stále směřoval k vyšším hodnotám funkcionálu a pokud to již nejde, tj. je nalezeno maximum funkcionálu, pak je třeba stisknout knoflík [**Maximum**].

V této scéně je možné si vygenerovat i neomezený mnohoúhelník. Proto je pro řízení výpočtu přidán ještě knoflík [**Neomezený**], který je třeba stisknout v okamžiku, kdy z vrcholu vychází hrana tvořená neomezenou polopřímkou, podle které hodnoty funkcionálu rostou. Kotouček nezčervená, protože jeho poloha nepředstavuje optimální polohu (ta neexistuje). Nemačkejte knoflík, pokud z obrázku vidíte, že hodnoty funkcionálu nejsou shora omezeny, ale není to patrno z vrcholu, ve kterém je kotouček.

Při pokynu jít nesprávným směrem a nebo předčasném ohlášení konce výpočtu knoflíky [**Maximum**] nebo [**Neomezený**] Algovize příkaz neprovede a protestuje. Chybová hlášení začínají vykřičníky a jsou následující:

- "Klesající směr" - v případě, že byl dán pokyn jít ve směru, ve kterém funkcionál klesá (bez ohledu na to, zda je hrana omezená úsečka nebo neomezená přímka.
- "Není maximum" - v případě, že byl stisknut knoflík [**Maximum**] v okamžiku, kdy kotouček není v místě maxima funkcionálu na mnohoúhelníku.
- "Neomezený směr" - v případě, že byl dán pokyn jít ve směru, ve kterém funkcionál roste podle neomezené hrany; v takovém případě je třeba stisknout knoflík [**Neomezený**].
- "Jen shora omezené" - v případě, že byl stisknut knoflík [**Neomezený**], ale hodnoty funkcionálu na všech hranách vycházejících z vrcholu označeného kotoučkem jsou shora omezené (což nastává když hrany jsou buď omezené úsečky bez ohledu na to, zda na nich funkcionál roste nebo klesá a nebo neomezené polopřímky, na nichž funkcionál směrem od vrcholu s kotoučkem klesá).

Nastavte si libovolně výchozí situaci a doveďte kotouček do vrcholu maximalizujícího funkcionál nebo do vrcholu, ze kterého vychází polopřímka, na které funkcionál není shora omezený. Opakujte výpočet z různých výchozích situací, dokud se vám několikrát za sebou nepodaří vše provést bez jediné chyby (tedy bez hlášení s vykřičníky).

Scéna: *Polyedr v prostoru*

V této scéně zopakujeme vše, co bylo výše řečeno pro dvojdimenzionální Euklidovský prostor, tedy pro rovinu, znovu pro třídimenzionální prostor tak, aby nečinilo potíže zobecnit to pro libovolnou konečnou dimenzi.

Rovnice $ax + by + cz = d$, která je obdobou rovnice přímky v rovině, je rovnice roviny ve 3D prostoru. Znázornit rovinu v prostoru na dvojdimenzionálním displeji není snadné a proto to dělat nebudeme.

Rovina rozděluje 3D prostor na dva podprostory stejně tak jako přímka rozděluje rovinu na dvě poloroviny. Tyto podprostory (chápány jako uzavřené, tedy zahrnující i hraniční rovinu) jsou v případě roviny s rovnicí $ax + by + cz = d$ dány nerovnostmi $ax + by + cz \leq d$ a $ax + by + cz \geq d$. Druhou budeme v zájmu jednotnosti zápisu psát ve formě $a'x + b'y + c'z \leq d'$, kde $a' = -a$, $b' = -b$, $c' = -c$ a $d' = -d$. Stejně jako rovinu, ani poloprostor nelze na dvojdimenzionálním displeji snadno znázornit.

Polyedr je průnik konečného množství poloprostorů. Je to konvexní těleso, jehož příklad je znázorněn v této scéně. Matematicky je to tedy množina bodů třídimenzionálního prostoru, která vyhovuje soustavě nerovností

$$\begin{aligned} a_1x + b_1y + c_1z &\leq d_1 \\ a_2x + b_2y + c_2z &\leq d_2 \\ &\dots \\ a_mx + b_my + c_mz &\leq d_m \end{aligned}$$

Polyedr jde ve většině případů na rovinném displeji znázornit snadno. Tato scéna nabízí volbu z několika připravených polyedrů. Na ovladači je volba mezi náhodným polyedrem, dokonalými Platonovskými tělesy a několika jehlany. Při volbě [**Náhodný**] je možno vygenerovat nový náhodně vytvořený polyedr knoflíkem [**Nový polyedr**], přitom lze zvolit vytvoření omezeného nebo neomezeného polyedru. V poli [**N=**] je možné nastavit požadovaný počet nerovností. Skutečný počet stěn polyedru ale bude obvykle jiný. Pokud je hodnota N příliš malá a je požadován *omezený* polyedr, často se stane, že N náhodných nerovností omezený polyedr nevytvoří a proto pak Algovize přidává další náhodné nerovnosti až do dosažení omezenosti. Na druhé straně se stává, že některé nerovnosti jsou zbytečné - jejich vynechání polyedr nezmění, což také znamená, že nevytvářejí žádnou novou stěnu. Takové nerovnosti nejsou ani zobrazovány.

Polyedrem na displeji můžeme pohybovat myší: jestliže je na ovladači zvolena volba [**Rotuj**], pak horizontální pohyb myši rotuje polyedrem kolem svislé osy a vertikální pohyb myši kolem vodorovné osy ležící v rovině obrazovky. Pokud je zvolena volba [**Pohybuj**], pak pohybem myši táhneme polyedr po obrazovce, aniž by se natáčel. Nakonec při volbě [**Zoomuj**] vertikální pohyb myši polyedr zvětšuje nebo zmenšuje bez natáčení (horizontální pohyb myši nemá na zobrazení vliv).

Poznamenejme, že polyedr může být omezený, ale také neomezený (volba [**Neomezený**]). Při otáčení neomezeného polyedru se pak může stát, že se díváme (z nekonečna) dovnitř; je tam tma a vidíme jen bíle naznačené hrany. Vnitřní body sice do polyedru patří, ale pro názornější zobrazení je polyedr zobrazen jako dutý, protože simplexový algoritmus pracuje výhradně na jeho povrchu. V případě omezeného polyedru předpokládáme, že pozorovatel je dostatečně vzdálen, takže zůstává stále vně polyedru.

Anatomie 3D polyedru je obdobná jako u konvexního mnohoúhelníka v rovině, ale složitější. Body, které splňují všechny nerovnosti definující polyedr jako ostré, jsou ve vnitřku polyedru a v třídimenzionálním prostoru je pozorovatel mimo polyedr nevidí. Body, které všechny nerovnosti splňují tak, že alespoň v jedné nerovnosti platí rovnost, jsou na povrchu neboli hranici polyedru. Všechny body polyedru, které splňují jistou nerovnost jako rovnost, vytvářejí *stěnu* polyedru. říkáme, že nerovnost a stěna si navzájem odpovídají. Mnohé pravidelné polyedry (dokonalá neboli Platonova tělesa) se označují nebo pojmenovávají počtem jejich stěn; například krychle je šestistěn, pravidelný trojboký jehlan je čtyřstěn, známe i osmistěn, dvanáctistěn a dvacetistěn. Pravidelná tělesa je také možno si zobrazit příslušnou volbou na ovladači.

Na displeji jsou také vypínatelně zobrazeny nerovnosti, které polyedr definují. Barva zde ale neoznačuje, zda je nerovnost splněna, ale nerovnosti odpovídající viditelným stěnám mají podklad stejné barvy jsou má odpovídající stěna. Nerovnosti, které odpovídají stěnám, které pro dané natočení polyedru nejsou viditelné, jsou napsány šedě na černém pozadí.

Hrana polyedru je úsečka, polopřímka nebo přímka, která je průnikem dvou sousedních stěn. Hrany omezeného polyedru jsou vždy úsečky, neomezený polyedr má alespoň některé hrany tvořeny polopřímkami. Přímka jako hrana se vyskytuje jen zřídka. *Vrchol* polyedru je bod, ve kterém se stýká více hran (obvykle 3 hrany, ale někdy - například u některých pravidelných těles - i více hran). Stejně tak je možno vrchol popsat jako bod, ve kterém se stýkají alespoň 3 stěny (a obvykle právě 3 stěny).

Scéna: *Nerovnosti*

Na rozdíl od roviny je ve třídimenzionálním prostoru nemožné kurzorem vybrat jednoznačně a srozumitelně libovolný bod v prostoru. Pokud ale zamíříte kurzorem na obrázek polyedru, vyberete tím jednoznačně viditelný bod polyedru, tedy jeho bod na přivráceném povrchu. Není možné vybrat bod pod povrchem (ve vnitřku polyedru) nebo na jeho odvrácené straně (pokud jím neotočíte). Ukazuje-li kurzor mimo obrázek polyedru, pak nevybírá žádný bod.

Pokud je knoflík myši stisknut, jsou barvy nerovností stejné jako v předchozí scéně, ale při uvolněném knoflíku jsou voleny jinak. Nerovnosti odpovídající neviditelným stěnám jsou sice zase psány šedě na černém podkladu, ale nerovnosti odpovídající viditelným stěnám jsou psány černým písmem na zeleném podkladu, který však může být buď tmavě nebo světle zelený.

Volba světle nebo tmavě zelené barvy ukazuje, jak jsou nerovnosti odpovídající viditelným stěnám splněny pro bod určený polohou kurzoru. Pohybujte kurzorem po displeji, aniž by byl stisknut knoflík myši. Je-li kurzor mimo obrázek polyedru, jsou všechny nerovnosti odpovídající viditelným stěnám světle zelené. Zajeďte nyní kurzorem na polyedr. Nerovnost, odpovídající stěně, ve které je kurzor, nyní bude tmavá.

V zásadě jsou 3 možnosti. Buď je kurzor uvnitř některé stěny; pak je tmavě zelená pouze jediná nerovnost, která odpovídá této stěně. Nebo se kurzor dostane na některou hranu, ale nikoli na její koncový bod. Jelikož hrana je společná dvěma stěnám, budou nyní tmavě zelené dvě nerovnosti, odpovídající stěnám, které spolu sdílejí uvedenou hranu. Poslední možnost je, že kurzor se dostane na vrchol polyedru. Vrchol obvykle patří najednou do právě tří stěn, takže tmavě zelené budou právě tři nerovnosti. V některých situacích ale vrchol patří do více stěn a proto je-li kurzor nad vrcholem, budou někdy tmavě zelené čtyři nebo i více nerovností. Zkuste si například zvolit libovolný vrchol dokonalého osmistěnu nebo dvacetistěnu nebo horní vrchol alespoň 4-bokého jehlanu.

Důležité je, že ukazuje-li kurzor na vrchol, tedy jsou-li alespoň tři nerovnosti tmavě zelené, pak zvolit hranu vycházející z tohoto vrcholu znamená vybrat z alespoň tří tmavých nerovností právě dvě, zbývající nerovnosti nechat zesvětlet. Hrana totiž patří do právě dvou stěn a je zvolena pokud se vyberou nerovnosti odpovídající těmto stěnám.

V obvyklém případě, kdy každý vrchol leží v právě 3 hranách (tj. z našich příkladů prakticky vždy u náhodného polyedru, dále u trojbokého jehlanu, krychle a dvanáctistěnu), jsou pro kurzor ve vrcholu právě tři nerovnosti tmavé a libovolný výběr dvou z nich (tři možnosti) určuje jednoznačně jednu ze tří hran vycházejících z vrcholu. Důležité je si uvědomit, že výběr dvou tmavých nerovností ze tří (tedy dvou stěn dotýkajících se vrcholu určeného kurzorem) je totéž jako *nevybrání* jedné (té zbývající) nerovnosti, resp. jedné stěny dotýkající se vrcholu.

Pokud z vrcholu vycházejí 4 nebo více hran, pak zjevně ne každý výběr dvou stěn dotýkajících se vrcholu určuje hranu (t.j. ne každé dvě stěny zahrnující vrchol mají společnou hranu).

Scéna: *Lineární funkcionál v prostoru*

Lineární funkcionál v třídimenzionálním prostoru je zobrazení, které každému bodu $[x, y, z]$ prostoru přiřazuje reálné číslo $ax + by + cz$, kde a, b a c jsou pevně daná reálná čísla určující funkcionál.

Neumím zobrazit na displeji názorným způsobem obecný lineární funkcionál tak, aby znázorňoval hodnoty funkcionálu ve všech bodech prostoru. Není to ale pro naše účely nutné: podobně, jak tomu bylo v rovině, i v prostoru simplexový algoritmus operuje výhradně na povrchu polyedru, na kterém se hledá maximum funkcionálu. Proto funkcionál budeme znázorňovat tak, že každý bod na povrchu polyedru obarvíme barvou, které v tepelné škále odpovídá hodnotě funkcionálu v uvažovaném bodě. Tepelná škála byla vysvětlena ve scéně o lineárních funkcionálech v rovině.

Ve volbě funkce na ovladači přibyla další funkce [**Změň funkcionál**]. Je-li zvolena, pohybujte kurzorem po obrazovce se stisknutým knoflíkem myši. Podobně jako jsme jindy rotovali kolem os polyedr, nyní pohybujeme pomyslným vektorem (a, b, c) (není nakreslen) určujícím funkcionál a odpovídajícím způsobem se mění i jeho hodnoty na povrchu polyedru.

Úloha lineární optimalizace se nyní dá popsat tak, že hledáme bod, který je ze všech bodů polyedru nejbělejší.

Technický problém nastává v okamžiku, kdy je polyedr neomezený, protože na něm nabývají hodnoty funkcionálu libovolně hodnot libovolně blízkých plus a/nebo minus nekonečnu, zatímco tepelná škála používaná pro vizualizaci jeho hodnot je konečná. Problém je vyřešen tak, že nekonečné stěny a hrany jsou oříznuty a z polyedru je zobrazena jen část, která ale obsahuje všechny vrcholy, omezené hrany a stěny a alespoň kus z každé neomezené stěny a hrany. Je zřejmé, že rostou-li (resp. klesají) hodnoty funkcionálu podél hrany směrem k místu, kde je uříznuta, rostly by dále až do nekonečna (resp. klesaly do minus nekonečna).

Znovu poznamenávám, že polyedr je kreslen jako dutý, protože veškerá činnost simplexového algoritmu se odehrává na jeho povrchu a vnitřní body by jen činily obrázek méně přehledným.

Scéna: *Pohyb po hranách polyedru*

Tato scéna ukazuje polyedr stejně jako scéna předchozí. Nyní ale zkusíme cestovat po povrchu polyedru tak, že budeme přecházet z jednoho vrcholu do druhého po hranách.

Kotoučkem pohybujeme tak, že zvolíme volbu [**Posuň kotouček**] a pak klepneme myší na libovolnou hranu, která vychází z vrcholu obsazeného kotoučkem (pokud není vidět, je třeba rotovat polyedrem, až se objeví). Pohybujte kotoučkem po polyedru a sledujte barvu pozadí nerovností.

Pokud sedí kotouček ve vrcholu, pak tento vrchol splňuje obvykle tři z nerovností určujících polyedr a tři nerovnosti jsou proto tmavě zelené. Vydá-li se po hraně, pak jedna z nerovností začne platit jako ostrá nerovnost (v naší tabulce se stane světle zelenou - sledujte to při animaci). Pohybuje se přitom po přímce, která je průnikem hraničních rovin těch dvou nerovností, které zůstávají tmavě zelené, zatímco u zbývající nerovnosti, která byla na počátku pohybu tmavě zelená, se rozdíl mezi pravou a levou stranou stále zvětšuje.

Podívejme se nyní na ostatní nerovnosti. U některých levá strana při pohybu kotoučku klesá a "platí stále více". Taková nerovnost by zůstala v platnosti, i kdyby se kotouček pohyboval až do nekonečna. Pokud ale alespoň v jedné nerovnosti levá strana roste a tedy rozdíl pravé a levé strany klesá, poklesne po jisté době na nulu - nerovnost začne být splněna jako rovnost. Jakmile nastane první takový okamžik, není již možné dále pokračovat, aniž by bod nevystoupil ven z polyedru. To znamená, že jsme se dostali do jiného vrcholu polyedru. Povšimněte si, že v něm opětovně platí ve 3 nerovnostech rovnost: 2 jsou ty, které určovaly použitou hranu a třetí je nově splněná rovnost, představující zarážku dalšího postupu.

Ve vyjímečných situacích se může stát, že alespoň některý vrchol leží na okraji 4 nebo více stěn (viz osmistěn a dvacetistěn v příkladech). Pokud v takovém vrcholu leží kotouček, jsou tmavě zelené alespoň 4 nerovnosti. Při pohybu po hraně všechny kromě dvou zesvětlí. Může se také stát, že při pohybu kotoučku nastane rovnost současně ve dvou nebo více nerovnostech jiných než ty, které určují hranu. Pak zase budou při zastavení kotoučku alespoň 4 nerovnosti tmavě zelené.

Scéna: *Simplexový algoritmus v prostoru*

Úloha lineárního programování v třídimenzionálním prostoru je nalézt takový bod nebo body polyedru, pro které daný lineární funkcionál nabývá maximální hodnoty. Jak již bylo řečeno výše, znamená to v naší grafické reprezentaci najít nejbělejší bod na povrchu polyedru. Tato scéna pracuje s omezenými polyedry, takže řešení vždy existuje.

Vytvořte si nejprve vhodný polyedr a pak, využívajíce různé volby funkce myši, si ho nastavte do vhodné polohy, nastavte podle svých požadavků lineární funkcionál a do zvoleného vrcholu polyedru vložte kotouček. Poté stiskněte knoflík [**Počítej**]. Algovize si provede výpočet, přepne do animačního módu a umožní vám výpočet krokovat nebo zobrazit v běhu a to i s případným vracením. Je také možno vypnout zobrazování funkcionálu a/nebo nerovností.

Simplexový algoritmus pracuje v prostoru v prakticky stejně, jako tomu bylo v rovině:

zvolíme si libovolný počáteční vrchol polyedru (zde zamlčuji, že nalézt pro danou soustavu nerovností nějaký vrchol polyedru jimi definovaného není někdy vůbec lehké, dokonce i sama otázka neprázdnosti takového polyedru může být obtížná; tento problém ale nebudu podrobněji rozebírat);
pokud ve směrech všech hran, vycházejících z vrcholu, funkcionál klesá nebo zůstává konstantní, výpočet končí a aktuální bod je optimální řešení problému, v opačném případě Algovize libovolně zvolí hranu vycházející z bodu, podél níž funkcionál roste, převede kotouček po zvolené hraně do jejího druhého konce a tento postup opakuje.

Když je nalezeno optimální řešení, kotouček zčervená.

Scéna: *Simplexový algoritmus - neomezený polyedr*

Scéna je stejná jako předchozí, ale pracuje výhradně s neomezenými polyedry. Pokud se stane, že narazíme během výpočtu na nekonečnou hranu, podél které hodnota funkcionálu roste, znamená to, že úloha nemá omezené řešení.

Scéna: *Simplexuj si sám v prostoru*

Scéna ukazuje totéž jako předchozí dvě scény - výpočet simplexového algoritmu. Kotoučkem ale nepohybuje na pokyn uživatele Algovize, ale uživatel sám a to stejným způsobem jako ve scéně "Pohyb po hranách polyedru". Algovize jen kontroluje postup výpočtu a při chybném kroku protestuje jako tomu bylo při uživatelském simplexování v rovině.