



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY

INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

DISTRIBUOVANÉ OPTIMALIZAČNÍ PROGRAMY

DISTRIBUTED OPTIMIZATION PROGRAMS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE

AUTHOR

Bc. Pavel Dvořák

VEDOUCÍ PRÁCE

SUPERVISOR

Ing. Jan Roupec, Ph.D.

BRNO 2016

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	Bc. Pavel Dvořák
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	Ing. Jan Roupec, Ph.D.
Akademický rok:	2015/16

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 diplomové práce:

Distribuované optimalizační programy

Stručná charakteristika problematiky úkolu:

Řada optimalizačních úloh je obtížně řešitelná pro svoji složitost nebo rozsáhlost. V těchto situacích může být vhodným přístupem jejich dekompozice na několik souvisejících úloh, které budou vzájemně komunikovat a které budou řešeny separátně. Kromě složitosti může být důvodem pro dekompozici i samotný charakter úlohy. Aby mohl být tento přístup realizován, je třeba navrhnout způsob vzájemné komunikace úloh včetně rozhraní a protokolu a do tohoto rámce zasadit vhodné řešiče.

Cíle diplomové práce:

1. Provedte teoretický rozbor principu DOP.
2. Popište rozhraní úloh pro začlenění do DOP.
3. Navrhněte a naprogramujte vzorové řešiče pro DOP.
4. Ověřte funkčnost a použitelnost na testovacích úlohách.

Seznam literatury:

Griva I., Nash S. G., Sofer A. (2009): Linear and Nonlinear Optimization. George Mason University, Fairfax, Virginia.

Popela P., Sklenář J., Matoušek R., Roupec J., Mrázková E. (2012): Advanced Decomposition Techniques Applied to DOP, Mendel Journal series, Vol.2012, No.1, pp.582-587

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2015/16

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D.
ředitel ústavu

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

ABSTRAKT

Diplomová práce se zabývá teoretickým rozбором distribuovaných optimalizačních programů (*dále už jen DOP*), praktickým návrhem a programovou realizací vzorového řešiče DOP a ověřením jejich funkčnosti a použitelnosti.

ABSTRACT

Master's thesis deals with the theory of distributed optimization programs (*next time just DOP*), the suggestion and programme sample solver DOP and verification of the functionality of DOP ideas, principles and system architecture.

KLÍČOVÁ SLOVA

Distribuované optimalizační programy, Simplexová metoda, matematické programování.

KEYWORDS

Distributed optimization programs, Simplex method, mathematical programming.

BIBLIOGRAFICKÁ CITACE

Dvořák, P. *Distribované optimalizační programy*, Brno, Vysoké učení technické v Brně, Fakulta strojního inženýrství. 2016, Vedoucí diplomové/bakalářské práce Ing. Jan Roupec, Ph.D.

PODĚKOVÁNÍ

Rád bych poděkoval panu Ing. Janu Roupčovi, PhD. za odborné a cenné rady, vstřícnost, ochotu a pomoc při vedení této diplomové práce.

V neposlední řadě bych chtěl poděkovat rodičům za podporu a trpělivost během celé doby studia.

ČESTNÉ PROHLÁŠENÍ

Prohlašuji, že tato práce je mým původním dílem, zpracoval jsem ji samostatně pod vedením Ing. Jana Roupce, PhD. a s použitím literatury uvedené v seznamu.

V Brně dne 27.2.2016

.....

Dvořák Pavel

OBSAH

1	ÚVOD	15
2	OPTIMALIZACE	17
2.1	Princip optimalizace.....	17
2.2	Využití optimalizačních metod	17
2.3	Rozdělení optimalizačních metod	18
3	LINEÁRNÍ PROGRAMOVÁNÍ	19
3.1	Historie lineárního programování	19
3.2	Lineární program.....	19
3.3	Simplexová metoda	20
3.3.1	Princip Simplexové metody	21
3.4	Dualita	25
3.5	Síťové úlohy	26
4	PRINCIP DOP.....	29
4.1	Základní syntaktické elementy	29
5	PROGRAMOVÁ REALIZACE DOP	33
5.1	Testovací úloha	33
6	ZÁVĚR.....	41
7	SEZNAM POUŽITÝCH ZDROJŮ	43
8	SEZNAM ZKRATEK.....	45
9	PŘÍLOHY	47

1 ÚVOD

Aplikace výpočetní techniky v současné době zasahují snad do všech oblastí lidské činnosti. Kromě mnoha jiných typů použití je stále významnější použití počítačů pro podporu rozhodovacích procesů. Zde hraje velmi velkou roli optimalizace. Zpočátku se optimalizační algoritmy a jejich aplikace musely omezovat na menší úlohy vzhledem k malé výpočetní rychlosti a malé kapacitě paměti. S postupem doby a pokrokem ve výkonu výpočetní techniky je však možné řešit stále složitější a rozsáhlejší úlohy, čímž se aplikace optimalizačních metod a technik rozšiřují do stále nových a nových oblastí. Kromě vlastních metod a technik hraje velmi významnou roli také tvorba modelů. Složitost úloh má dvě stránky – jednak se jedná o vlastní komplikovanost související s (algoritmickou) obtížností řešeného problému, jednak se jedná o velikost úlohy vyjádřenou velikostí datového popisu (počet proměnných, omezení, náhodné vlivy, ...). V obou případech může řešitelnost úlohy na stávajících počítačích příznivě ovlivnit dekompozice úloh, která může omezit datovou velikost úlohy jejím rozkladem na sadu úloh menších s výrazně kratší dobou výpočtu, nebo také může za jistých okolností i umožnit paralelní zpracování těch částí úloh, které na sobě nejsou přímo závislé. Zajímavou metodou, jak toho dosáhnout, jsou distribuované optimalizační programy (DOP), které přináší v teoretické rovině [4], v rovině možností praktické realizace potom [3]. Tato práce má za úkol naprogramovat některé dílčí části a ověřit možnosti praktické realizace DOP.

2 OPTIMALIZACE

Optimalizace je matematická disciplína, která se zabývá nalezením nejlepšího řešení daného problému. Např. v případě vedení podniku optimalizace řeší úkoly typu maximalizace zisku, minimalizace ztrát, maximalizace efektivity a minimalizace rizika. V případě navrhování mostu to může být minimalizování zatížení a maximalizování nosnosti mostu. U letového plánu zase minimalizace času a spotřeby paliva a podobně. Nalezení optimálního řešení je natolik užitečné, že se rozšířilo téměř do každé oblasti použití. Optimalizační metody byly použity dokonce pro vysvětlení přírodních zákonů, jako třeba odvození Fermatova zákona lomu světla. [1]

2.1 Princip optimalizace

V principu se jedná o hledání extrému **účelové funkce**, tedy takových hodnot nezávisle proměnných, pro které účelová funkce nabývá maxima či minima. Ten je hledán na uzavřeném intervalu, jelikož optimalizační úlohy mívají daná omezení vyplývající z řešených problémů. Tento interval odpovídá definici *Weierstrassovy věty*: „Spojitá funkce je na uzavřeném intervalu ohraničená a nabývá zde svého absolutního maxima a absolutního minima.“. V tomto intervalu se nalezne optimální řešení, které obsahuje co nevyšší možné hodnoty pozitiv (*zisků*), které se snažíme maximalizovat a co nejnižší možné hodnoty negativ (*ztrát*), které chceme minimalizovat. Extrém funkce ale často nelze najít za pomoci metod matematické analýzy, a to kvůli tvaru matematického modelu optimalizační úlohy. Proto vznikly různé optimalizační metody, řešící různé typy úloh. Mnohé spadají do takzvaného *matematického programování*, některé využívají *heuristické metody* založené na *evolučních algoritmech* a některé z nich jsou založené na *algoritmech umělé inteligence*, využívajíc kupříkladu *neuronové sítě*. [3]

2.2 Využití optimalizačních metod

Optimalizační modely jsou využívány již po staletí, zejména pro svoji efektivitu. V současné době se stala optimalizace téměř nepostradatelnou, jelikož obchod, podnikání, firmy, byznys i jiná odvětví se neustále rozrůstají a stávají se čím dál komplikovanějšími. V řadě případů je téměř nemožné, nebo ekonomicky neefektivní, provést rozhodnutí, bez pomoci těchto modelů. V nadnárodních korporacích může zlepšení byť malého procenta operací vést k navýšení zisku řádově i o několik miliard. Bez optimalizačních metod by bylo dnes prakticky nemožné, navrhnu novou počítačovou čip, zahrnující miliony tranzistorů. [1]

Řešení takto složitých a jemných problémů je samozřejmě zásluhou pokroku, který během posledních několika desetiletí nastal v oblasti počítačového hardwaru a softwaru. Díky výpočetní technice dnešní doby je možno řešit úlohy dříve téměř nemyslitelné. Snad právě proto v posledních letech bývá optimalizace často zařazována i do informatiky. Nyní jsme schopni řešit problémy s tisíci, nebo dokonce miliony proměnných. Tím se optimalizační modely staly praktickým nástrojem jak v podnikání, tak ve vědě a technice. [1]

2.3 Rozdělení optimalizačních metod

Optimalizační úlohy se dělí dle *tvaru matematického modelu* a *charakteru* řešené úlohy. Jedním z nejdůležitějších rozdělení úlohy je na **deterministické**, které mají pevně dané veličiny a **stochastické**, které obsahují nejméně jednu náhodnou proměnnou veličinu a rozdělení pravděpodobnosti všech náhodných proměnných je známo. Členění dle typu problému a tedy i způsobu řešení, je rozděleno například do různých odvětví *matematického programování* [3]:

- *lineární programování,*
- *nelineární programování,*
- *celočíselné programování,*
- *parametrické programování,*
- *kvadratické programování,*
- *konvexní programování,*
- *dynamické programování,*
- *vícekritériální programování, atd.*

3 LINEÁRNÍ PROGRAMOVÁNÍ

Lineární programování (LP) se zabývá problémy souvisejícími s hledáním vázaných extrémů lineárních funkcí více proměnných, jejichž omezující podmínky mají tvar lineárních rovnic a nerovností. Účelová funkce i její omezení jsou tedy lineární kombinací prvků vektoru x . Ačkoliv jsou lineární funkce relativně jednoduché funkce, často se objevují v ekonomice, plánování, sítích, plánování výroby a dalších aplikacích. [2]

3.1 Historie lineárního programování

Řešením problémů lineárního programování se zabýval již v letech 1826-1888 teoretický fyzik J. B. J. Fourier (1888), v souvislosti s analytickou mechanikou a teorií pravděpodobnosti. Jeho myšlenky dále zpracoval Farkas (1902) počátkem 20. století. Ve třicátých letech našeho století byly v ekonomice řešeny lineární optimalizační problémy kombinatoricky, kupříkladu přiřazovací problém (Köning a Egerváry), ve třicátých a čtyřicátých letech dopravní problém (např. Hitchcock 1941). Rozvoj efektivních metod lineárního programování v jeho nynější podobě (J. von Neumann 1937, Kantorovič 1939) dovršil G. B. Dantzig (1949), který syntézou myšlenek předchozích autorů, včetně již zmíněného Fouriera, vytvořil tzv. *simplexovou metodu*, která je univerzálním nástrojem k řešení problémů lineárního programování. V současné době probíhají snahy o další zefektivnění řešení úloh lineárního programování, jako například *elipsoidová metoda*, navazující na práce Chačijanovy (1979), *Karmarkarova metoda* (Karmarkar 1984) a další. S rozvojem a nárůstem výkonu výpočetní techniky jsou zkoumány možnosti využití paralelních výpočtů řešící úlohy lineárního programování (Mangasarin O. L., De Leone R. 1986). [2]

3.2 Lineární program

Lineární program lze definovat jako úlohu pro minimalizaci účelové funkce v oblasti vyhraněné omezujícími podmínkami, kde účelová funkce i podmínky jsou lineární. Omezení lineárního programu mohou být formulovány ve tvaru rovností i rovností [3].

Dle metod řešení úloh lineárního programování rozlišujeme tři tvary úlohy lineárního programování, a to na minimalizační úlohu (maximalizační úlohu lze převést na úlohu minimalizační) lineárního programování ve tvaru:

- rovnic,
- nerovnic,
- smíšeném.

Úlohu lineárního programování ve tvaru nerovnic nebo smíšeném lze převést na tvar rovnicový přidáním pomocných přídatných (doplňkových) proměnných, které usnadní výpočet úlohy a zároveň neovlivní hodnotu ani polohu účelové funkce, jelikož koeficienty těchto proměnných jsou v účelové funkci nulové. [3]

Pro všechny tvary lineárního programování je množina přípustných řešení vždy uvnitř intervalu oblasti, která je ohraničena omezujícími podmínkami dané úlohy. [3]

Dále platí *základní věta lineárního programování*: „Má-li úloha lineárního programování optimální řešení, pak lze jeho optimální řešení najít mezi základními řešeními.“ [2]

Lineární program lze modifikovat například na úplně celočíselný lineární program (ILP), smíšený celočíselný lineární program (MILP), či bivalentní programování [3].

Dalším typem úloh nelineárního programování bývají *úlohy s nedělitelnostmi*, ve kterých se výsledek běžných metod lineárního programování zaokrouhluje na celá čísla. To souvisí s praktickými úlohami, které počítají s celočíselnými proměnnými, například osobami. Výsledné hodnoty jsou pro potřeby těchto úloh uspokojující. [3]

Celočíselné programování umožňuje řešit nejen problémy s nedělitelnostmi, ale také řadu komplikovaných problémů, které nelze jinými postupy řešit efektivně, nebo dokonce vůbec. Jedná se zejména o kombinatorické problémy, v nichž jde o nalezení optimálního řešení z konečné, i když obvykle velmi rozsáhlé, množiny přípustných řešení. Např. problém obchodního cestujícího nebo rozvrhovací problémy. [3]

3.3 Simplexová metoda

Simplexová metoda je nejpoužívanější metoda lineárního programování a jeden z nejpoužívanějších numerických algoritmů. Vznikla kolem roku 1940, současně v době nasazování modelů lineárního programování v ekonomice a vojenském plánování [1]. Vyvinul ji americký matematik *George Dantzig (1914 – 2005)* s využitím myšlenek *Jordanovy modifikace Gaussovy eliminační metody pro řešení lineárních algebraických soustav* a poprvé publikoval v roce 1947 [2]. V té době měla dosti konkurenčních metod, ale ty se nakonec ukázaly být méně efektivní a byly brzy zavrhnuty. Dokonce jak se problémy stávaly většími a počítače výkonnějšími, simplexová metoda se dokázala adaptovat a zůstala volbou pro mnoho lidí. Pouze v posledních letech s rozvojem metod vnitřního bodu měla vážného vyzyvatele o prvenství v oblasti lineárního programování. [1]

Přestože metoda řeší pouze úlohy lineárního programování, její techniky jsou použitelné i na další úlohy. Stejně techniky mohou být použity pro zpracování lineárních omezení v nelineárních optimalizačních problémech, a dají se zobecnit ke zpracování nelineárních omezení. Způsoby, kterými jsou omezení zastoupena, jsou použity v jiném uspořádání, jako jsou metody pro výpočet Lagrangeových multiplikátorů (dvojitý/duální proměnné). [1]

Simplexová metoda má významné historické vazby k ekonomii, které ovlivnily terminologii spojenou s metodou. Například, je běžné mluvit o snižování „nákladů“ a sledování „ceny“. Pro mnoho aplikací jsou tyto termíny užitečnou a sugestivní interpretací, která je uvedena v modelech lineárního programování. [1]

3.3.1 Princip Simplexové metody

Je to iterativní metoda, pro řešení úlohy lineárního programování, zapsané standardní formou. Pohybujeme se od jednoho základního možného řešení (extrémního bodu) k dalšímu. Jedná se o sofistikovaný způsob procházení krajních bodů, tedy základních řešení úlohy lineárního programování, množiny přípustných řešení. Z množiny přípustných řešení v uzavřeném intervalu, který nám ohraničují omezení úlohy, se hledání omezí na vcelku nízký počet krajních bodů. Z počátečního krajního bodu se přechází na sousední krajní bod, ve kterém je hodnota účelové funkce větší, pro případ maximalizace, či naopak menší, pro případ minimalizace. Pokud takový bod neexistuje, bylo nalezeno optimální řešení, nebo je některá hrana vycházející z tohoto krajního bodu neomezená, a pak optimální řešení neexistuje vzhledem k neomezenosti hodnoty účelové funkce na množině přípustných řešení. Existuje i možnost situace, kdy řešením je celá hrana, v takovémto případě je však za optimální řešení prohlášen první nalezený bod této hrany, viz *Základní věta lineárního programování*. Algoritmus je navržen pro výpočet pomocí počítače, neboť ruční výpočet je pro problémy s více proměnnými dosti komplikovaný, obsáhlý a časově náročný. [2]

Simplexová metoda se dá popsat geometrickou analogií, a to následovně: Necht' existuje předpoklad, že je znám krajní bod x_0 množiny přípustných řešení M . Z tohoto krajního bodu vychází konečné množství hran množiny M , z nichž každá buď obsahuje jediný další krajní bod této množiny M , nebo je neomezená. Jestliže na některé neomezené hraně existuje bod, pro který je hodnota účelové funkce větší než $c^T x_0$, nemá úloha optimální řešení a postup končí. V opačném případě je hledán sousední krajní bod, pro který je hodnota kriteriální funkce větší než $c^T x_0$. Necht' je to krajní bod x_1 . Pak stejný postup, který byl dosud aplikován na bod x_0 , bude aplikován na bod x_1 . Pokud neexistuje sousední krajní bod s vlastností $c^T x > c^T x_0$, je x_0 hledaným optimálním řešením. [2]

V případě nedegenerované úlohy končí po konečném počtu kroků buď nalezením optimálního řešení, nebo zjištěním, že optimální řešení neexistuje. Takzvaná finitnost je zaručena existencí konečného množství krajních bodů, které jsou vyšetřovány v pořadí, ve kterém jejich hodnoty kriteriální funkce rostou, tedy každý z nich nanejvýš jednou. Přestože by byla možná konstrukce příkladu, ve kterém by bylo úkolem vyšetřit všechny krajní body, jejich počet je většinou relativně malý.

Jelikož každá maximalizační úloha lineárního programování lze převést na úlohu ve tvaru rovnic, zkonstruuujeme tedy simplexovou metodu v tomto speciálním případě úlohy lineárního programování ve tvaru rovnic, neboli v *kanonickém tvaru úlohy lineárního programování*:

Hledejme maximum kriteriální funkce

$$z = c_1x_1 + c_2x_2 + \dots + c_nx_n \quad (3.1)$$

a při splnění omezujících podmínek (vzájemně lineárně nezávislých)

$$\begin{aligned} x_1 &+ a_{1,m+1}x_{m+1} + \dots + a_{1,k}x_k + \dots + a_{1,n}x_n = b_1 \\ x_2 &+ a_{2,m+1}x_{m+1} + \dots + a_{2,k}x_k + \dots + a_{2,n}x_n = b_2 \\ &\vdots \\ x_m &+ a_{m,m+1}x_{m+1} + \dots + a_{m,k}x_k + \dots + a_{m,n}x_n = b_m \end{aligned} \quad (3.2)$$

a při splnění podmínek nezápornosti

$$x_1 \geq 0, x_2 \geq 0, \dots, x_n \geq 0. \quad (3.3)$$

Přitom c_j ($j = 1, 2, \dots, n$), a_{ij} ($i = 1, 2, \dots, m; j = 1, 2, \dots, n$), b_i ($i = 1, 2, \dots, m$) jsou daná reálná čísla. Necht' je předpoklad, že $b_i \geq 0$ pro všechna $i = 1, 2, \dots, m$, pokud nebude uvedeno jinak. Dosazením za každou z proměnných $x_{m+1}, x_{m+2}, \dots, x_n$ hodnotou rovnou nule, dostaneme výchozí základní řešení

$$x_0^T = (b_1, b_2, \dots, b_m, 0, 0, \dots, 0), \quad (3.4)$$

kde posledních $m - n$ složek tohoto vektoru nabývá hodnoty 0. Tím je splněno, že přípustné základní řešení je soustava sloupců $\mathbf{a}_i$ matice $\mathbf{A}$, odpovídajících nenulovým x_i , lineárně závislá. Pak je možno tyto proměnné eliminovat (vyjádřit pomocí pravých stran). Eliminované proměnné v soustavě lineárně nezávislých rovnic $\mathbf{Ax} = \mathbf{b}$ jsou nazývány *bázickými proměnnými*. Jejich počet je roven m . Ostatní proměnné x_i se nazývají *nebázickými proměnnými*. Ty jsou rovny nule. [2]

V případě nede degenerované úlohy hledání optimálního řešení probíhá ve formě určitého počtu iterací. Každá iterace znamená přechod od jednoho základního řešení, které je ekvivalentem některému krajnímu bodu množiny přípustných řešení, k základnímu řešení odpovídajícímu sousednímu krajnímu bodu, v němž hodnota kritériální funkce roste. Každá iterace je transformací lineárních algebraických rovnic na ekvivalentní soustavu, která je také ve tvaru kanonickém. V této transformaci se jedna nebázická proměnná stane bázickou, vstoupí do báze a anuluje se, takže počet m bázických proměnných zůstane v každé iteraci zachován. Pokud jsou bázické proměnné $x_1, x_2, \dots, x_m$ doplňkovými proměnnými maximalizačního problému, vyjádřeného původně ve tvaru nerovnic, hodnota kritériální funkce odpovídající výchozímu základnímu řešení je ve tvaru

$$z(x_0) = 0 \cdot b_1 + 0 \cdot b_2 + \dots + 0 \cdot b_m + c_{m+1} \cdot 0 + \dots + c_n \cdot 0 = 0. \quad (3.5)$$

Zatímco ve výchozím řešení jsou obecně nenulové proměnné u nulových koeficientů a nulové proměnné u nenulových koeficientů, lze očekávat, že výsledná iterace bude obsahovat výraz s největší hodnotu $c_{m+1}x_{m+1} + \dots + c_nx_n$. Toto výchozí základní řešení je jedno z přípustných řešení popisované úlohy. [2]

Pokud se řešení změní tak, že některé dosud nebázické x_k bude namísto nuly obsahovat hodnotu λ , pro kterou platí $\lambda > 0$. V této iteraci je index k roven některému z čísel $m+1, m+2, \dots, n$. Tím se vektor řešení změní na nový, vyplývající ze soustavy (7):

$$x_1^T = (b_1 - \lambda a_{1k}, b_2 - \lambda a_{2k}, \dots, b_m - \lambda a_{mk}, 0, \dots, 0, \lambda, 0, \dots, 0), \quad (3.6)$$

kde hodnota λ odpovídá k -té složce vektoru x_1^T . Dosazením tohoto řešení do kritériální funkce (6), dostaneme

$$z(x_1) = c_1(b_1 + \lambda a_{1k}) + c_2(b_2 + \lambda a_{2k}) + \dots + c_m(b_m + \lambda a_{mk}) + c_k \lambda. \quad (3.7)$$

Po jednoduché úpravě dostaneme

$$\begin{aligned} z(x_1) &= c_1 b_1 + c_2 b_2 + \dots + c_m b_m - \lambda(c_1 a_{1k} + c_2 a_{2k} + \dots + c_m a_{mk} - c_k) = \\ &= c_1 b_1 + c_2 b_2 + \dots + c_m b_m + c_k \lambda - c'_k \lambda, \end{aligned} \quad (3.8)$$

kde

$$c'_k = c_1 a_{1k} + c_2 a_{2k} + \dots + c_m a_{mk} \quad (3.9)$$

Pokud je interpretována hodnota $x_i (i = 1, 2, \dots, n)$ jako „úroveň i -tého procesu“, pro cesty vztažené k základním proměnným jako „základní procesy“, a hodnotu kritériální funkce z (6) jako zisk z procesů $x_1, x_2, \dots, x_n$ při splnění omezujících podmínek (7), které se dají interpretovat jako „zdrojová omezení“, pak c_k je zvýšení zisku způsobené zvednutím jednotkové úrovně procesu x_k . Současně c'_k je snížení zisku ze základních procesů (tj. v této první iteraci z procesů $x_1, x_2, \dots, x_m$), způsobené tímto zavedením, při dodržení zdrojových omezení. [2]

Celkový přírůstek kritériální funkce, způsobený zvýšením hodnoty původně nebázické proměnné x_k o λ , při respektování zdrojových omezení, je roven

$$z(x_1) - z(x_0) = -\lambda \Delta_k, \quad (3.10)$$

kde

$$\Delta_k = c'_k - c_k \quad (3.11)$$

je *úbytek* z , způsobený zvýšením hodnoty proměnné x_k z 0 na 1 při respektování omezujících podmínek. Pokud je úbytek záporný, má samozřejmě význam přírůstek [2].

Dle hodnoty veličiny Δ_k lze snadno zjistit, které nebázické proměnné by bylo účelné zvýšit hodnotu, aby se respektování zdrojových omezení zvýšila hodnota kritériální funkce. Pokud by byla zařazena do báze u proměnné x_k , pak by platilo, s podmínkou $\lambda > 0$, pro potenciální přírůstek kritériální funkce následující:

$$\begin{aligned} z(x_1) - z(x_0) &> 0, & \text{je-li } \Delta_k < 0 \\ z(x_1) - z(x_0) &= 0, & \text{je-li } \Delta_k = 0 \\ z(x_1) - z(x_0) &< 0, & \text{je-li } \Delta_k > 0 \end{aligned} \quad (3.12)$$

Tento vztah bývá označován jako *kritérium optimality*. [2]

Je patrné, že v případě řešení *maximalizačního* problému je x_0 optimálním řešením právě tehdy, když platí $\Delta_j \geq 0$ pro kterékoliv $j = 1, 2, \dots, n$. Výběrem jiného bázeického řešení x_1 by se hodnota kritériální funkce nezlepšila, jelikož dle *kritéria optimality* (3.12) by pro každé

$k = 1, 2, \dots, n$ platilo $z(x_1) - z(x_0) \leq 0$. To znamená, že již nelze nalézt proměnnou x_k , jejíž zařazení do bazického řešení by zvýšilo hodnotu kritériální funkce. [2]

Naopak, *minimalizační* úloha má optimální řešení právě tehdy, platí-li $\Delta_j \leq 0$ pro všechna $j = 1, 2, \dots, n$ [2].

Jestliže v maximalizačním problému platí $\Delta_j < 0$, pak dosáhneme zvýšení kritériální funkce tím, že zařadíme proměnnou x_j do základního řešení. [2]

Nejrychlejší docílení optima je zvolením nové bazické proměnné, pro kterou je záporní hodnota Δ_j nejmenší. Tato proměnná se nazývá *vstupní proměnná* x_k a určí se podle vztahu:

$$\min \Delta_j = \Delta_k \quad (3.13)$$

za podmínky, že Δ_k je záporné. k -tý sloupec matice strukturálních koeficientů A , jehož index je roven indexu vstupní proměnné, je nazýván *klíčový sloupec* [2] (v publikacích v anglickém jazyce označován jako *pivot column*). [1]

Pokud se v dané iteraci nedojde k optimálnímu řešení, je vybrána dle (3.13) vhodná proměnná do nového zadání. Dále je nutno vybrat proměnnou, která se ze základního řešení vyřadí, a určit hodnotu nové bazické proměnné. V případě, že za proměnnou x_k dosadíme hodnotu $\lambda > 0$, základní řešení změní svoji podobu. Původní bazické proměnné v (3.2) pak nabývají hodnot:

$$x_1 = b_1 - \lambda a_{1+k}, x_2 = b_2 - \lambda a_{2+k}, \dots, x_m = b_m - \lambda a_{m+k} \quad (3.14)$$

Kvůli neporušení podmínek nezápornosti musí platit:

$$\begin{aligned} b_1 - \lambda a_{1+k} &\geq 0 \\ b_2 - \lambda a_{2+k} &\geq 0 \\ &\vdots \\ b_m - \lambda a_{m+k} &\geq 0. \end{aligned} \quad (3.15)$$

Když $a_{ik} \leq 0$, pak $(b_1, b_2, \dots, b_m)^T \geq \mathbf{0}^T$, omezení (3.15) je splněno pro libovolné $\lambda > 0$. Za x_k je tedy možno dosadit hodnotu rostoucí nade všechny meze. [2]

Je-li však $a_{ik} > 0$, platí úprava:

$$\lambda \leq \frac{b_i}{a_{ik}} \quad (i = 1, 2, \dots, m; \ a_{ik} > 0). \quad (3.16)$$

Pro neporušení podmínek nezápornosti nesmí hodnota proměnné λ , nově zaváděné do báze, překročit hodnotu tohoto podílu. Aby podmínka platila pro všechna i , musí platit i pro i , pro něž je b_i/a_{ik} minimální. Při transformaci řešení nesmí hodnota λ překročit minimální z kladných podílů b_i/a_{ik} , takže

$$\lambda \leq \min \frac{b_i}{a_{ik}} = \frac{b_l}{a_{lk}} \quad (3.17)$$

Pro $a_{ik} > 0, 0 \in \{1, 2, \dots, m\}$. Je-li za vstupní proměnnou x_k vybrána hodnota

$$\lambda = \frac{b_l}{a_{lk}}, \quad (3.18)$$

pak bázecká proměnná, jejíž strukturní koeficient v l -tém řádku je nenulový a roven jedné, nabývá hodnoty nulové, a tedy přestává být bázeckou proměnnou, neboť dle vztahů před (3.15) platí

$$x_l = b_l - \frac{b_l}{a_{lk}} a_{lk} = 0. \quad (3.19)$$

Její původní hodnota, rovna b_l , se tedy anulovala. Proměnnou x_l lze tedy považovat za vyloučenou ze základního řešení a je nazývána proměnnou *vystupující*. V základním řešení je tato proměnná nahrazena vstupující proměnnou x_k . l -tý řádek matice strukturních koeficientů $\mathbf{A}$ a vektoru kapacitních limitů $\mathbf{b}$, který je vybrán minimalizací dle (3.17), je nazýván *klíčovým řádkem*. k -tý sloupec a l -tý řádek definují tzv. *klíčový prvek* a_{lk} , který je důležitým členem při transformaci rovnic (3.2) na efektivní soustavu s cílem přechodu k lepšímu základnímu řešení. [2]

Transformace je provedena tak, aby ekvivalentní soustava byla rovněž v kanonickém tvaru, tj. aby po anulování nebázeckých proměnných byla každá bázecká proměnná přímo vyjádřena pomocí pravé strany soustavy rovnic. Jelikož pořadí sčítanců na levé straně rovnic (3.2) je libovolně zaměnitelné, bázecké proměnné po transformaci soustavy nemusí být umístěny v prvních m sloupcích matice $\mathbf{A}$ tak, jak tomu je ve speciálním případě ve vztahu (3.2), ale mohou být umístěny v libovolných m sloupcích matice $\mathbf{A}$. Bázecká proměnná x_{l_i} ($i = 1, 2, \dots, m$) pak přísluší l_i -tému sloupci matice $\mathbf{A}$, jestliže vyjádření její hodnoty pomocí pravé strany, tj. $x_{l_i} = b_i$, je umožněno rovnicí vzniklé z i -tého řádku vztahu $\mathbf{Ax} = \mathbf{b}$. Neboť bázecká proměnná x_{l_i} přísluší i -tému řádku a l_i -tému sloupci soustavy $\mathbf{Ax} = \mathbf{b}$. [2]

3.4 Dualita

S každou úlohou lineárního programování je úzce spojena jiná úloha, která je rovněž lineární, a která je původní úlohou jednoznačně určena. Přitom maximalizační úloze odpovídá úloha minimalizační a naopak. Úlohy lineárního programování se tedy vyskytují ve dvojicích *duálně sdružených úloh*. Ty se dělí na *primární* (původní) úlohu a *duální*, tedy úlohu s ní sdruženou. Dualita není jevem pouze lineárního programování, ale je to jev obecnější. Mezi duálně sdruženými úlohami platí několik vztahů, které jsou užitečné jak praxi, tak z teoretického hlediska. [2]

Velké využití nachází například v *ekonomice*, kde vyrábějící firma může duální úlohu využít jako úlohu prodeje výrobků s odlišnou marží pro odlišné zákazníky či obchodní řetězce. Rovněž nalézá využití v úlohách *analýzy citlivosti*. Za určitých okolností bývá dokonce i výhodnější než simplexová metoda. Jde například o *duální simplexovou metodu*, *primárně*

duální metodu a metodu MODI, využívanou především pro řešení lineárních dopravních úlohách. [2]

Duální úloha je tvořena výlučně koeficienty primární úlohy. Dvojice *symetricky duálně sdružených úloh* je jakýmsi „zrcadlovým obrazem“.

$$\begin{array}{ll}
 f(x) = c^T x \rightarrow \max & g(u) = b^T x \rightarrow \min \\
 Ax \leq b & A^T u \geq c \\
 x \geq 0 & u \geq 0
 \end{array} \tag{3.20}$$

Existuje možnost, že výchozí úloha vypadá jinak, než výše uvedené vztahy. V maximalizační úloze se mohou objevit rovnice, nebo nerovnice typu $\geq$ a některé proměnné mohou být záporné, nebo neomezené co do znaménka. V takovýchto případech je možné postupovat následujícími kroky: transformujeme výchozí optimalizační úlohu do tvaru (3.1) – (3.2) a poté „zrcadlově převrácenými“ zpětnými úpravami zjednodušíme odpovídající duální úlohu do výsledného tvaru. [2]

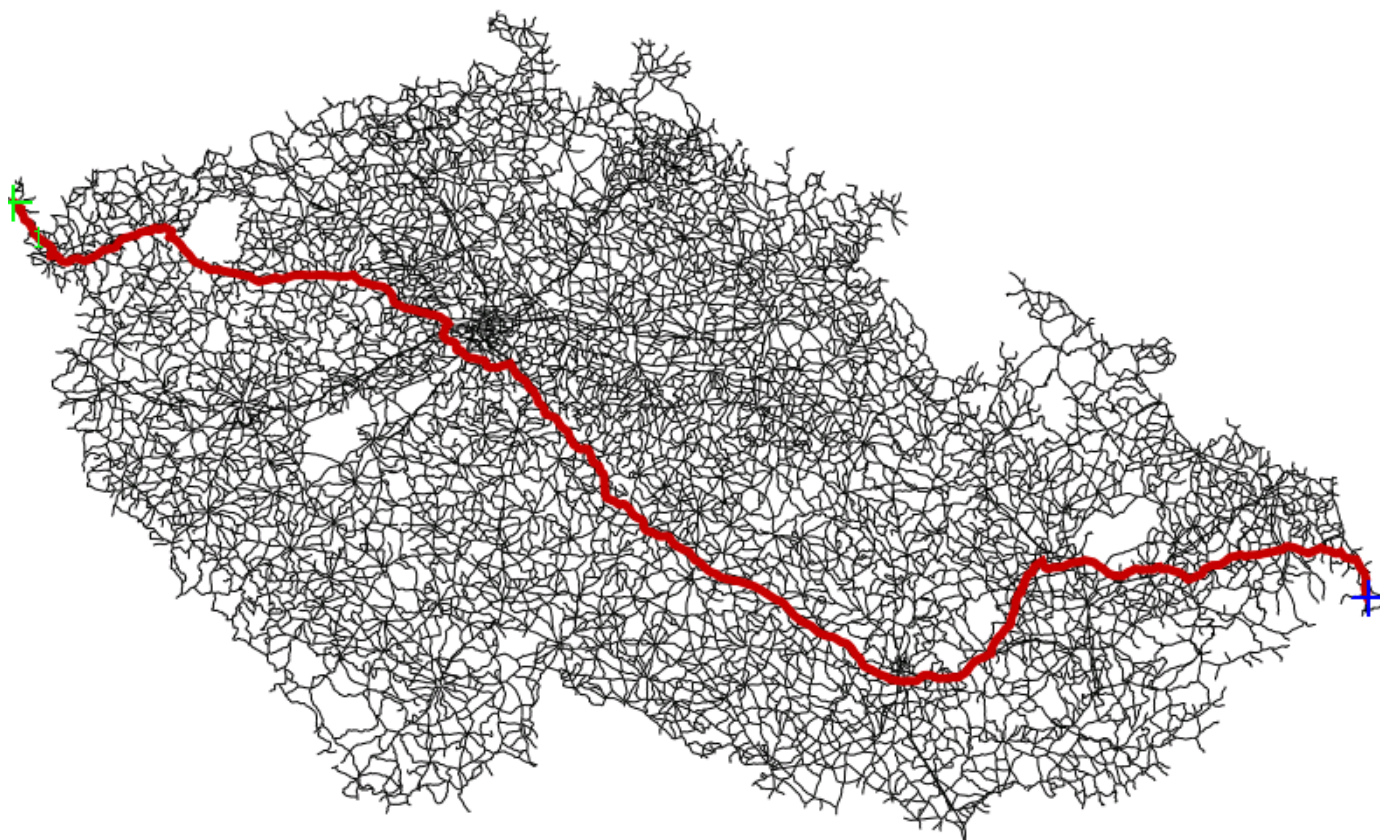
Jelikož duální úloha je úlohou lineárního programování, je ji možno řešit např. simplexovou metodou. Pokud jsme však simplexovou metodou vyřešili již primární úlohu, není nutné duální úlohu již řešit. Je totiž současně nalezeno řešení duální úlohy. Toto tvrzení platí i naopak, tedy řešením duální úlohy je získáno řešení úlohy primární. [2]

3.5 Síťové úlohy

Lineární programy mohou být definovány na sítích. Matematický program je popsán sítí, která je tvořena orientovaným grafem, který má m uzlů a n hran. Tok hranou označme prvkem vektoru $\mathbf{x}$, který je pro lepší orientaci indexován dvojicí indexů, jelikož hrana spojuje dva uzly. Tedy prvek $x_{i,k}$ reprezentuje tok hranou vedoucí z uzlu i do uzlu k . Prvek $c_{i,k}$ reprezentuje cenu za jednotkový tok touto hranou. Orientovaný graf, reprezentujícího síť, lze vyjádřit formou matice $\mathbf{A}$ o rozměrech $m \times n$, kde řádky reprezentují uzly a sloupce hrany. Pokud je prvek $a_{ij} = -1$, hrana j vychází z uzlu i , tedy hranou z uzlu něco odtéká. Pokud je $a_{ij} = +1$, hrana j vstupuje do uzlu i , tedy v tomto uzlu končí. Pokud je $a_{ij} = 0$, hrana j nemá žádnou incidenci s uzlem i . Prvky vektoru $\mathbf{b}$ reprezentují tok mezi uzlem a vnějším prostředím. Nulová hodnota znamená, že uzel je čistě přechodný. Kladná hodnota znamená odtok z uzlu ze sítě. Záporná hodnota přítok do uzlu z vnějšího prostředí. Soustava rovnic omezení $\mathbf{Ax} = \mathbf{b}$ vyjadřující podmínku, že pro každý uzel musí být součet všech přítoků a odtoků roven nule. [3]

Síťové úlohy mohou reprezentovat jak abstraktních úloh, tak i reálných řešení sítí elektrického vedení, dopravních, potrubních, datových, apod. – proto je řešení těchto úloh velice praktické a užitečné. Nalézají uplatnění při řešení problémů v dopravě, logistice, směrování dat atd. Síťové úlohy mohou být řešeny pomocí modifikované simplexové metody, algoritmy z oblasti teorie grafů a heuristickými metodami (pro rozsáhlé úlohy). [3]

Na obrázku 1 je ukázka síťové úlohy nalezení nejrychlejší cesty z nejvýchodněji položené obce v ČR (Hrčava) do nejzápadněji položené obce (Krásná) v systému GRASS.



Obr 3.1) Ukázka síťové úlohy.

4 PRINCIP DOP

Optimalizační přístupy se uplatňují stále v nových oblastech. To je umožněno pokrokem ve výpočetní technice, který umožňuje řešit úlohy stále složitější a rozsáhlejší. Složitost optimalizační úlohy může být posuzována ze dvou stran – kvantitativní a kvalitativní. Kvantitativní složitost spočívá v rozsahu úlohy daném velikostí dat (počet proměnných, parametrů, omezení), Kvalitativní stránka se potom vztahuje ke struktuře úlohy [3].

Úlohy velkého rozsahu často vznikají spojováním původně nezávislých menších úloh v prostoru nebo v čase. Tyto úlohy mohou být řešeny jako celek použitím běžných metod a algoritmů, aniž by bylo využito jejich struktury k usnadnění řešení. Tento přístup „řešení hrubou silou“ je založen na extenzivním přístupu, tedy využití rostoucí výpočetní rychlosti a paměťové kapacity počítačů. Další pokrok lze dosáhnout například paralelizací výpočtu tam, kde je to možné (typicky např. maticové operace) a kde jsou k dispozici odpovídající technické prostředky. Ačkoliv se technické limity neustále posouvají, je výhodné při řešení využívat i speciální strukturu rozsáhlých úloh. [3]

Řešení složitých reálných optimalizačních úloh vyžaduje používání komplikovaných matematických modelů. Pro účely zápisu modelu úlohy pro nějaký konkrétní softwarový produkt je vhodné mít k dispozici nástroje, které umožní jednoduché zacházení s modelem a jeho modifikaci v průběhu řešení. Pro řešení určitých typů úloh může být vhodným nástrojem dekompozice, ovšem podobné přístupy, používané pro rozmanité typy optimalizačních úloh, jsou obvykle využívány izolovaně případ od případu. V [4] je představen jednotící rámec založený na grafické reprezentaci a objektovém přístupu, který je nazvaný Distribuované optimalizační programy (Distributed optimization programme – DOP). Jak je v originálním pramenu [4] uvedeno, má se jednat o výraznou pomoc zejména ve stádiu modelování (manipulace s matematickými programy) a méně již ve stádiu samotného řešení. Protože se ale i při řešení často používá distribuovaný přístup, dekompozice a paralelní výpočty, je vhodné uvažovat o propojení obou fází a aplikovat DOP i na fázi řešení. Způsob, jak toho lze dosáhnout, je uveden v kapitole 5.

4.1 Základní syntaktické elementy

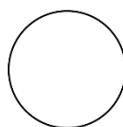
Popis základních elementů v duchu [4] lze nalézt např. v [5], [6] nebo [3], na jejich základě je sestaven popis v této kapitole. Ze zkušenosti lze obecně usuzovat, že statické programy mohou být chápány jako souhrn optimalizačních elementů majících určitou strukturu:

$$\mathcal{O} = (C, F, G), \quad (4.1)$$

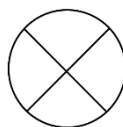
kde symboly C , F a G definují interní strukturu optimalizačních elementů. C představuje rozhodovací proměnnou $\mathbf{x}$ a příslušnou množinu přípustných řešení C . F je funkce pro vyčíslení

elementu; pro optimalizační element je to obvykle účelová funkce $f(\mathbf{x})$. Symbol G potom představuje cíl pro rozhodnutí $\mathbf{x}$, tedy např. $\mathbf{x} \in \operatorname{argmin}\{\dots\}$.

Pro vizualizaci struktury optimalizační úlohy a jejích elementů byla vyvinuta jednoduchá grafická reprezentace. Optimalizační elementy O jsou symbolizovány uzlem grafu. Ten je zakreslen pomocí kolečka, uvnitř kterého se nachází doplňující symbol upřesňující typ daného elementu. Příklady jsou uvedeny na obrázcích obr. 4.1 až obr. 4.4. Protože práce [4] je zaměřena na stochastické programování, obsahuje mnoho specializovaných elementů pro tuto oblast, zejména celou řadu elementů pro deterministické ekvivalenty úloh stochastického programování. Tyto druhy elementů nejsou pro další text podstatné, proto v následujícím přehledu nejsou zahrnuty.

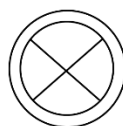


Obr. 4.1) Obecný optimalizační element.

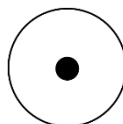


Obr. 4.2) Random element.

Náhodný element na obr. 4.2 představuje zdroj náhodných hodnot při modelování stochastických úloh; např. náhodný element R může být specifikován pomocí C , které zahrnuje náhodný vektor s odpovídajícím rozdělením pravděpodobnosti, F může být definováno např. jako očekávaná hodnota, cíl G v tomto případě nebude specifikován.



Obr. 4.3) Základní matematický program stochastické úlohy.



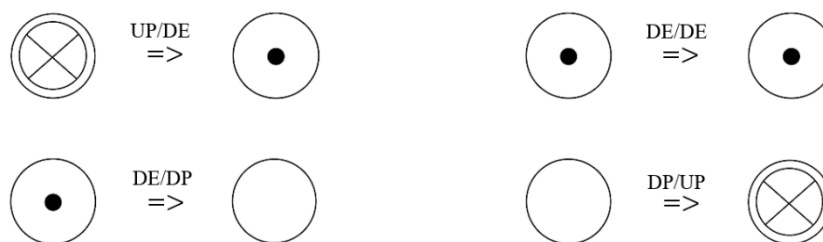
Obr. 4.4) Deterministický ekvivalent stochastického programu.

S grafy reprezentujícími matematické programy (resp. modely optimalizačních úloh) lze provádět syntaktické transformace (dle [4] *element trace*), kdy jednotlivé elementy mohou být dekomponovány na několik elementů s jistými vazbami; to je ekvivalentem dekompozice úloh matematického programování. Pro tuto transformaci je používán operátor $\Rightarrow$.

Každý optimalizační element v grafu obecně reprezentuje matematický program. Mezi dvěma programy (elementy) reprezentujícími programy $\mathcal{C}_1$, $\mathcal{C}_2$, pokud $\mathcal{C}_1$ reprezentuje

nadřazený uzel („master“) a $\mathcal{C}_2$ reprezentuje podřazený uzel („slave“), může docházet k těmto variantám vzájemného vztahu [4, 3]:

1. $\mathcal{C}_1$ a $\mathcal{C}_2$ se nijak neovlivňují, tzn. že v grafu nejsou spojeny žádnou hranou.
2. $\mathcal{C}_1$ modifikuje strukturu $\mathcal{C}_2$ ovlivňováním C_2 a F_2 . Současně existuje „zpětná vazba“ modifikací F_1 vyvolaného $\mathcal{C}_2$. V grafu se pro znázornění tohoto vztahu používá jednoduchá šipka směřující od $\mathcal{C}_1$ k $\mathcal{C}_2$. Tato situace je znázorněna na obr. 4.5.
3. $\mathcal{C}_1$ ovlivňuje $\mathcal{C}_2$ stejně jako v předchozí variantě, v opačném směru ale k žádnému ovlivnění nedochází (řešení $\mathcal{C}_2$ nijak neovlivňuje uzel $\mathcal{C}_1$). V grafu se tento vztah znázorňuje pomocí dvojité šipky směřující od $\mathcal{C}_1$ k $\mathcal{C}_2$. Tento vztah je označován jako „silná vazba“.
4. $\mathcal{C}_1$ nijak neovlivňuje $\mathcal{C}_2$, ale musí čekat na výsledek $\mathcal{C}_2$, neboť tento výsledek modifikuje F_1 . Tento vztah se v grafu označuje čárkovaná šipka a nazývá se slabá vazba.

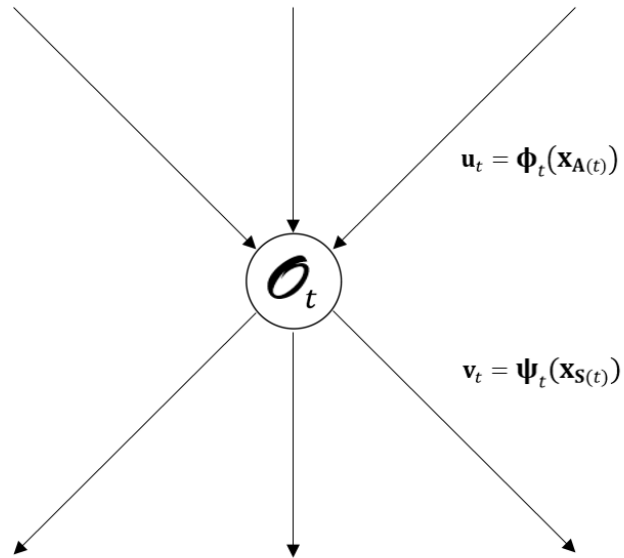


Obr. 4.5) Vazba mezi elementy

Kromě optimalizačních a random elementů jsou zavedeny ještě tzv. „transformační elementy“. Zjednodušeně lze říci, že jejich existence souvisí se skutečností, že za jistých okolností nelze mezi uzly předávat data „tak, jak jsou“. Jednotlivé uzly totiž mohly vzniknout dekompozicí původní úlohy (resp. původního uzlu) pomocí syntaktických transformací. Do uzlu následujícího obvykle nemůže být předána úloha z uzlu předcházejícího bez provedení jisté transformace, neboť ve své původní podobě by v cílovém uzlu neměla smysl (tedy např. ve výše uvedené situaci 3 nemůže být $\mathbf{x}_1^{opt}$ přímo dosazeno za C_2). Všechny tři uvedené typy elementů jsou souhrnně označovány jako DOP elementy. Podrobnosti lze nalézt např. v [3].

S transformačními elementy souvisejí transformační funkce Φ a Ψ . Pro napojení prvku $\mathcal{C}_t$ na ostatní prvky je nutné specifikovat hodnoty $\mathbf{u}_t$ (vstup do prvku $\mathcal{C}_t$) a $\mathbf{v}_t$ (výstup z uzlu $\mathcal{C}_t$), viz obr. 4.6. Pokud je (A_t) množina indexů DOP elementů, které jsou bezprostředně předcházející elementu $\mathcal{C}_t$, pak $\mathbf{u}_t = \Phi_t(\mathbf{x}_{A(t)}) = \Phi_t((\mathbf{x}_a)_{a \in A(t)})$, tedy vstup do elementu $\mathcal{C}_t$ vznikne transformací výstupů (výsledků) elementů bezprostředně předcházejících. Podobně pro

elementy bezprostředně následující elementu $\mathcal{O}_t$ použijeme indexy z množiny $S(t)$, a pak lze psát $\mathbf{v}_t = \Psi_t(\mathbf{x}_{S(t)}) = \Psi((\mathbf{x}_s)_{s \in S(t)})$.



Obr. 4.6) Interakce DOP elementů

5 PROGRAMOVÁ REALIZACE DOP

Smyslem práce bylo realizovat některé vzorové řešiče a opatřit je vhodným rozhraním pro snadné začlenění do systému DOP připravovaného na FSI [3]. Systém předpokládá princip, kdy datový popis, obsahující celou strukturu úlohy včetně všech vstupních dat a omezení, má k dispozici pouze řídicí program (master), který dle situace a konfigurace vybírá vhodné řešiče, předává jim vstupní data, spouští je a přebírá jejich data výstupní. Řešiče jsou optimalizačními elementy ve smyslu terminologie kapitole 4, master potom kromě řídicí role zastává i funkce transformačních elementů. Optimalizační elementy nemají přímo komunikovat mezi sebou navzájem, ale pouze prostřednictvím DOP master programu. Pro komunikaci mezi masterem a ostatními uzly je předpokládáno použití TCP protokolu (protože bude nad TCP vytvořena nadstavba pro komunikaci optimalizačních uzlů, tedy vlastně nový protokol aplikační vrstvy TCP/IP modelu, je v [3] tento protokol označen jako OTP – Optimization Transfer Protocol). Vzhledem k tomu, že v době zpracování této práce nebyla realizace DOP master úlohy ani knihovny pro podporu OTP ještě zcela dokončena, bylo nutné pracovat v simulovaném prostředí. Proto byla naprogramována provizorní jednoúčelová úloha simulující DOP master program, aby bylo možné ověřit spouštění řešičů. Výměna dat mezi master úlohou a řešiči probíhá prostřednictvím textových souborů, ovšem programová realizace je provedena tak, aby bylo možné snadno začlenit použití OTP protokolu, jakmile bude k dispozici.

Všechny programy bylo naprogramovány v programovacím jazyce *C# .NET* ve *Visual Studiu 2015* od společnosti *Microsoft*. S ohledem na architekturu systému ale na použitém programovacím jazyce nezáleží, neboť jednotlivé elementy jsou tvořeny samostatnými spustitelnými programy, jediným omezením by pak mohla být schopnost použít (zavolat) funkce pro komunikaci pomocí OTP, které mají být uloženy v knihovně *dll*.

5.1 Testovací úloha

Jako testovací úloha byla zvolena úloha lineárního programování, pro její řešení byl naprogramován řešič, který využívá simplexovou metodu (viz 3.5). Obecný tvar úlohy lineárního programování pro případ maximalizace (např. [3]) je

$$? \in \arg \max_{\mathbf{x}} \{z = \mathbf{c}^T \mathbf{x} \mid \mathbf{Ax} \diamond \mathbf{b}, \mathbf{l} \leq \mathbf{x} \leq \mathbf{u}\}, \quad (5.1)$$

kde:

$z = \mathbf{c}^T \mathbf{x}$ je účelová funkce,

$\mathbf{c} = (c_j) \in \mathbb{R}^n$, $j \in \mathcal{J}$, jsou koeficienty účelové funkce, $\mathcal{J}$ je množina indexů proměnných, obvykle se volí $\mathcal{J} = \{1, \dots, n\}$,

$\mathbf{b} = (b_i) \in \mathbb{R}^m$, $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{m \times n}$, $i \in \mathcal{I}$, $j \in \mathcal{J}$, jsou koeficienty omezujících podmínek, $\mathcal{I}$ je množina indexů pro omezující podmínky, obvykle $\mathcal{I} = \{1, \dots, m\}$,

$\mathbf{l} = (l_j) \in \mathbb{R}^n$, $\mathbf{u} = (u_j) \in \mathbb{R}^n$, $j \in J$ jsou vektory horních a dolních pro $\mathbf{x}$ (obvykle je pro jednotlivé prvky vektoru $\mathbf{x}$ dolní mez rovna 0 a horní mez ∞ , meze také bývají často zahrnuty již přímo v omezeních),

$\diamond$ je symbol reprezentující příslušný relační operátor, $\diamond \in \{\leq, \geq, =\}^m$.

Pro naši testovací úlohu je zadaná účelová funkce

$$Z = 59,9x_1 + 47,4x_2 + 59,7x_3 + 52,9x_4 + 11,3x_5 + 16,4x_6 + 11,6x_7 + 12,5x_8$$

a omezující podmínky

$$2,3x_1 + 1,9x_2 + 1,7x_3 + 1,7x_4 < 83$$

$$1,3x_1 + 1,3x_2 + 2,2x_3 + 2,7x_4 < 97$$

$$1,5x_1 + 2,3x_2 + 2,6x_3 + 1,6x_4 < 91$$

$$1,2x_1 + 2,0x_2 + 1,3x_3 + 2,7x_4 < 86$$

$$1,5x_1 + 1,6x_2 + 2,2x_3 + 2,4x_4 < 86$$

$$2,3x_1 + 1,9x_2 + 1,7x_3 + 1,7x_4 + 4,3x_5 + 3,9x_6 + 3,8x_7 + 3,2x_8 < 129$$

$$1,3x_1 + 1,3x_2 + 2,2x_3 + 2,7x_4 + 3,6x_5 + 3,1x_6 + 3,7x_7 + 3,4x_8 < 134$$

$$1,5x_1 + 2,3x_2 + 2,6x_3 + 1,6x_4 + 4,3x_5 + 4,1x_6 + 4,5x_7 + 3,6x_8 < 154$$

$$1,2x_1 + 2,0x_2 + 1,3x_3 + 2,7x_4 + 4,3x_5 + 4,5x_6 + 4,3x_7 + 3,6x_8 < 123$$

$$1,5x_1 + 1,6x_2 + 2,2x_3 + 2,4x_4 + 3,2x_5 + 3,2x_6 + 4,3x_7 + 4,1x_8 < 140,$$

tedy v maticovém tvaru:

$$\begin{bmatrix} 2,3 & 1,9 & 1,7 & 1,7 & 0,0 & 0,0 & 0,0 & 0,0 \\ 1,3 & 1,3 & 2,2 & 2,7 & 0,0 & 0,0 & 0,0 & 0,0 \\ 1,5 & 2,3 & 2,6 & 1,6 & 0,0 & 0,0 & 0,0 & 0,0 \\ 1,2 & 2,0 & 1,3 & 2,7 & 0,0 & 0,0 & 0,0 & 0,0 \\ 1,5 & 1,6 & 2,2 & 2,4 & 0,0 & 0,0 & 0,0 & 0,0 \\ 2,3 & 1,9 & 1,7 & 1,7 & 4,3 & 3,9 & 3,8 & 3,2 \\ 1,3 & 1,3 & 2,2 & 2,7 & 3,6 & 3,1 & 3,7 & 3,4 \\ 1,5 & 2,3 & 2,6 & 1,6 & 4,3 & 4,1 & 4,5 & 3,6 \\ 1,2 & 2,0 & 1,3 & 2,7 & 4,3 & 4,5 & 4,3 & 3,6 \\ 1,5 & 1,6 & 2,2 & 2,4 & 3,2 & 3,2 & 4,3 & 4,1 \end{bmatrix} \begin{matrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{matrix} * < \begin{bmatrix} 83 \\ 97 \\ 91 \\ 86 \\ 86 \\ 129 \\ 134 \\ 154 \\ 123 \\ 140 \end{bmatrix}$$

Vzhledem ke tvaru omezení (nulová oblast v „severovýchodní“ oblasti matice **A**) lze původní úlohu rozdělit na dvě úlohy, přičemž první z nich bude mít pouze čtyři proměnné. Toto rozdělení může být prakticky zdůvodnitelné např. tím, že se jedná o maximalizaci zisku dosaženého ve výrobním procesu, který se odehrává ve dvou etapách, a v první etapě vstupují do procesu pouze proměnné x_1 až x_4 . Pro první etapu je tedy nutné řešit úlohu, jejíž účelová funkce má tvar

$$Z_1 = 59,9x_1 + 47,4x_2 + 59,7x_3 + 52,9x_4$$

a omezující podmínky jsou

$$2,3x_1 + 1,9x_2 + 1,7x_3 + 1,7x_4 < 83$$

$$1,3x_1 + 1,3x_2 + 2,2x_3 + 2,7x_4 < 97$$

$$1,5x_1 + 2,3x_2 + 2,6x_3 + 1,6x_4 < 91$$

$$1,2x_1 + 2,0x_2 + 1,3x_3 + 2,7x_4 < 86$$

$$1,5x_1 + 1,6x_2 + 2,2x_3 + 2,4x_4 < 86.$$

Ve druhé etapě se potom se řeší upravená původní úloha, kde hodnoty pro proměnné x_1 až x_4 jsou již známy z etapy první. V účelové funkci se tedy již tyto proměnné neobjevují a v omezeních je nutné odečíst zdroje spotřebované v první etapě. Tedy účelová funkce bude mít tvar:

$$Z_2 = 11,3x_5 + 16,4x_6 + 11,6x_7 + 12,5x_8$$

a omezující podmínky budou

$$2,3x_1 + 1,9x_2 + 1,7x_3 + 1,7x_4 + 4,3x_5 + 3,9x_6 + 3,8x_7 + 3,2x_8 < 129$$

$$1,3x_1 + 1,3x_2 + 2,2x_3 + 2,7x_4 + 3,6x_5 + 3,1x_6 + 3,7x_7 + 3,4x_8 < 134$$

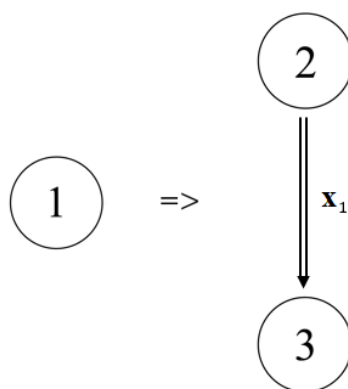
$$1,5x_1 + 2,3x_2 + 2,6x_3 + 1,6x_4 + 4,3x_5 + 4,1x_6 + 4,5x_7 + 3,6x_8 < 154$$

$$1,2x_1 + 2,0x_2 + 1,3x_3 + 2,7x_4 + 4,3x_5 + 4,5x_6 + 4,3x_7 + 3,6x_8 < 123$$

$$1,5x_1 + 1,6x_2 + 2,2x_3 + 2,4x_4 + 3,2x_5 + 3,2x_6 + 4,3x_7 + 4,1x_8 < 140,$$

kde se ovšem za x_1 až x_4 dosadí optimální hodnoty získané v první etapě a nerovnice se upraví.

Z pohledu principů DOP popsaných v kapitole 4 se jedná o situaci, kdy jedna optimalizační úloha vyjádřená jedním optimalizačním elementem se syntaktickou transformací převede na dva spolupracující elementy se silnou vazbou mezi nimi. Situace je znázorněna na obr. 5.1.



Obr. 5.1) Znázornění transformace testovací úlohy

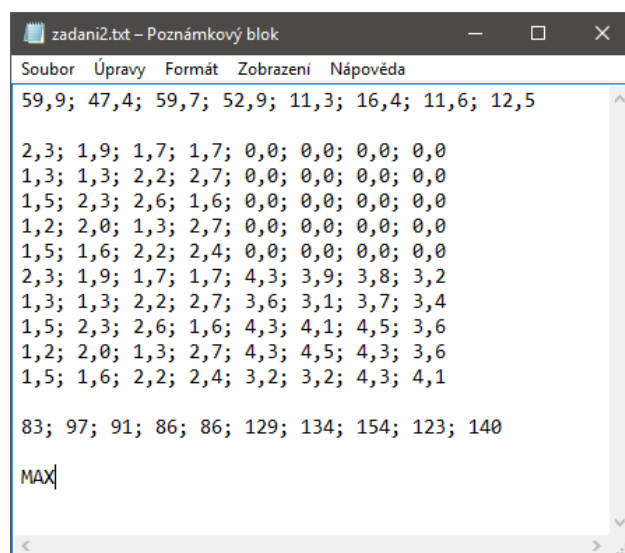
Vzhledem k charakteru úlohy (důsledkem je silná vazba mezi elementy) neovlivňuje uzel 3 uzel 2, a proto je potřebná pouze jedna transformace dat, a sice pro data proudící z uzlu 2 do uzlu 3. Tato transformace (z pohledu elementu 3 se v souladu s terminologií kapitoly 4 označená jako Φ_3) zahrnuje dosazení řešení z uzlu 2 do omezujících podmínek úlohy druhé

etapy (element 3). Pro účely programové realizace plní úlohu transformačního elementu master úloha. Pro obě etapy se spouští totožný řešič, pokaždé ale dostane od master úlohy odlišná data.

Řešič metody simplex je naprogramován jako program *SimplexMethod*. Program předpokládá úlohu lineárního programování ve tvaru rovnic nebo nerovnic (operátor <, >, =), pro úpravu na rovnicový tvar je použita metoda přídatných proměnných; tuto operaci provádí řešič automaticky a není nutné ji nijak zvenčít obsluhovat. Matice pak vypadá takto:

$$\begin{bmatrix}
 2,3 & 1,9 & 1,7 & 1,7 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\
 1,3 & 1,3 & 2,2 & 2,7 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\
 1,5 & 2,3 & 2,6 & 1,6 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\
 1,2 & 2,0 & 1,3 & 2,7 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\
 1,5 & 1,6 & 2,2 & 2,4 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\
 2,3 & 1,9 & 1,7 & 1,7 & 4,3 & 3,9 & 3,8 & 3,2 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 & 0,0 \\
 1,3 & 1,3 & 2,2 & 2,7 & 3,6 & 3,1 & 3,7 & 3,4 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 & 0,0 \\
 1,5 & 2,3 & 2,6 & 1,6 & 4,3 & 4,1 & 4,5 & 3,6 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 & 0,0 \\
 1,2 & 2,0 & 1,3 & 2,7 & 4,3 & 4,5 & 4,3 & 3,6 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0 & 0,0 \\
 1,5 & 1,6 & 2,2 & 2,4 & 3,2 & 3,2 & 4,3 & 4,1 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 1,0
 \end{bmatrix}
 \cdot
 \begin{bmatrix}
 x_1 \\
 x_2 \\
 x_3 \\
 x_4 \\
 x_5 \\
 x_6 \\
 x_7 \\
 x_8 \\
 x_9 \\
 x_{10}
 \end{bmatrix}
 =
 \begin{bmatrix}
 83 \\
 97 \\
 91 \\
 86 \\
 86 \\
 129 \\
 134 \\
 154 \\
 123 \\
 140
 \end{bmatrix}$$

Program *Master* má za úkol zpracovat zadání z textového souboru (Obr. 5.2), kde je úplné zadání celé úlohy, předpokládá se tvar nerovnic (operátor <). Master poté rozčlení zadání na menší dílčí úlohy. Poté proběhne optimalizace první etapy (prvního dne), kde *Master* předá upravené zadání a spustí program simplexové metody *SimplexMethod*, pomocí kterého získáme optimální řešení první etapy (Obr. 5.3) a výsledné hodnoty jsou předány zpět programu *Master* (Obr. 5.5).



Obr. 5.2) Zadání úlohy v textovém souboru

Poté je proveden přepočítání omezení (zásob) a jsou odečteny zásoby spotřebované v první etapě. Tímto je nachystáno zadání druhé etapy, které je opět předáno programem *Master* programu *SimplexMethod*, který je jím opět spuštěn a výpočtem simplexové metody je získáno optimální řešení druhé etapy (Obr. 5.4) a předáno zpět programu *Master* (Obr. 5.5).

```

C:\Users\DVORAKPA.COMIMPEX\Desktop\Master\Master\bin\Debug\SimplexMethod.exe
2,30 1,90 1,70 1,70 1,00 0,00 0,00 0,00 0,00 83,00
1,30 1,30 2,20 2,70 0,00 1,00 0,00 0,00 0,00 97,00
1,50 2,30 2,60 1,60 0,00 0,00 1,00 0,00 0,00 91,00
1,20 2,00 1,30 2,70 0,00 0,00 0,00 1,00 0,00 86,00
1,50 1,60 2,20 2,40 0,00 0,00 0,00 0,00 1,00 86,00
59,90 47,40 59,70 52,90 0,00 0,00 0,00 0,00 0,00 0,00
Z = 0

1,00 0,83 0,74 0,74 0,43 0,00 0,00 0,00 0,00 36,09
0,00 0,23 1,24 1,74 -0,57 1,00 0,00 0,00 0,00 50,09
0,00 1,06 1,49 0,49 -0,65 0,00 1,00 0,00 0,00 36,87
0,00 1,01 0,41 1,81 -0,52 0,00 0,00 1,00 0,00 42,70
0,00 0,36 1,09 1,29 -0,65 0,00 0,00 0,00 1,00 31,87
0,00 -2,08 15,43 8,63 -26,04 0,00 0,00 0,00 0,00 -2161,61
Z = 2161,60869565217
x_0 = 36,0869565217391

1,00 0,30 0,00 0,50 0,76 0,00 -0,50 0,00 0,00 17,81
0,00 -0,66 0,00 1,33 -0,02 1,00 -0,83 0,00 0,00 19,45
0,00 0,71 1,00 0,33 -0,44 0,00 0,67 0,00 0,00 24,72
0,00 0,71 0,00 1,68 -0,34 0,00 -0,28 1,00 0,00 32,48
0,00 -0,42 0,00 0,93 -0,17 0,00 -0,73 0,00 1,00 4,89
0,00 -13,06 0,00 3,54 -19,30 0,00 -10,34 0,00 0,00 -2542,99
Z = 2542,98833819242
x_0 = 17,8134110787172
x_2 = 24,7230320699708

1,00 0,52 0,00 0,00 0,85 0,00 -0,11 0,00 -0,53 15,21
0,00 -0,06 0,00 0,00 0,23 1,00 0,21 0,00 -1,43 12,47
0,00 0,86 1,00 0,00 -0,38 0,00 0,93 0,00 -0,35 22,99
0,00 1,46 0,00 0,00 -0,03 0,00 1,04 1,00 -1,80 23,68
0,00 -0,45 0,00 1,00 -0,19 0,00 -0,79 0,00 1,07 5,25
0,00 -11,48 0,00 0,00 -18,63 0,00 -7,56 0,00 -3,80 -2561,58
Z = 2561,58448060075
x_0 = 15,2127659574468
x_2 = 22,9943679599499
x_3 = 5,24718397997497

=====
x[0] = 15,2127659574468
x[1] = 0
x[2] = 22,9943679599499
x[3] = 5,24718397997497
Vysledne reseni Z(optimalni) = 2561,58448060075

```

Obr. 5.3) Výpočet první etapy

```

C:\Users\DVORAKPA.COMIMPEX\Desktop\Master\Master\bin\Debug\SimplexMethod.exe
4,30 3,90 3,80 3,20 1,00 0,00 0,00 0,00 0,00 46,00
3,60 3,10 3,70 3,40 0,00 1,00 0,00 0,00 0,00 49,47
4,30 4,10 4,50 3,60 0,00 0,00 1,00 0,00 0,00 63,00
4,30 4,50 4,30 3,60 0,00 0,00 0,00 1,00 0,00 60,68
3,20 3,20 4,30 4,10 0,00 0,00 0,00 0,00 1,00 54,00
11,30 16,40 11,60 12,50 0,00 0,00 0,00 0,00 0,00 0,00
Z = 0

1,10 1,00 0,97 0,82 0,26 0,00 0,00 0,00 0,00 11,79
0,18 0,00 0,68 0,86 -0,79 1,00 0,00 0,00 0,00 12,90
-0,22 0,00 0,51 0,24 -1,05 0,00 1,00 0,00 0,00 14,64
-0,66 0,00 -0,08 -0,09 -1,15 0,00 0,00 1,00 0,00 7,61
-0,33 0,00 1,18 1,47 -0,82 0,00 0,00 0,00 1,00 16,26
-6,78 0,00 -4,38 -0,96 -4,21 0,00 0,00 0,00 0,00 -193,44
Z = 193,435897435898
x_1 = 11,7948717948718

=====
x[0] = 0
x[1] = 11,7948717948718
x[2] = 0
x[3] = 0
Vysledne reseni Z(optimalni) = 193,435897435898

```

Obr. 5.4) Výpočet druhé etapy

Výsledné optimální řešení je součtem optimálních řešení obou etap, tedy $Z = Z_1 + Z_2$ (viz Obr. 5.5).

```

C:\Users\DVORAKPA.COMIMPEX\Desktop\Master\Master\bin\Debug\Master.exe
=====
* * * * * DOP1 * * * * *
=====
Proběhl výpočet Simplexové metody.
=====
x[1] = 15,2127659574468
x[2] = 0
x[3] = 22,9943679599499
x[4] = 5,24718397997497
=====
výsledek DOP1 = 2561,58448060075
=====

* * * * * DOP2 * * * * *
=====
Proběhl výpočet Simplexové metody.
=====
x[1] = 0
x[2] = 11,7948717948718
x[3] = 0
x[4] = 0
=====
výsledek DOP2 = 193,435897435898
=====

Výsledek DOP: Z(optimalni) = 2755,02037803665
=====

Ukončete stisknutím klávesy Enter.

```

Obr. 5.5) Výpis výsledných hodnot DOP programem Master

Demonstrace funkčnosti DOP je na Obr. 5.6, kde je porovnán výsledek simplexové metody z Obr. 5.7 a výsledek DOP z Obr. 5.5. Jak je vidět, oba výsledky jsou shodné. Výsledky testovacích úloh byly kontrolovány softwarem pro matematické programování a optimalizaci GAMS (*The General Algebraic Modeling System*) a hodnoty byly téměř totožné.

SimplexMethod	Master
<pre> 0,00 -0,45 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 1,10 1,00 0,00 -0,06 0,00 0,00 0,18 0,00 0,00 0,00 0,00 0,00 -0,22 0,00 0,00 1,46 0,00 0,00 -0,66 0,00 0,00 0,00 0,00 0,00 -0,33 0,00 0,00 -11,48 0,00 0,00 -6,78 0,00 Z = 2755,02037803665 x_0 = 15,2127659574468 x_2 = 22,9943679599499 x_3 = 5,24718397997497 x_5 = 11,7948717948718 ===== x[0] = 15,2127659574468 x[1] = 0 x[2] = 22,9943679599499 x[3] = 5,24718397997497 x[4] = 0 x[5] = 11,7948717948718 x[6] = 0 x[7] = 0 Výsledné řešení Z(optimalni) = 2755,02037803665 </pre>	<pre> ===== x[1] = 15,2127659574468 x[2] = 0 x[3] = 22,9943679599499 x[4] = 5,24718397997497 ===== výsledek DOP1 = 2561,58448060075 ===== * * * * * DOP2 * * * * * ===== Proběhl výpočet Simplexové metody. ===== x[1] = 0 x[2] = 11,7948717948718 x[3] = 0 x[4] = 0 ===== výsledek DOP2 = 193,435897435898 ===== Výsledek DOP: Z(optimalni) = 2755,02037803665 ===== </pre>

Obr. 5.6) Srovnání výsledných hodnot programů SimplexMetohod a Master

```
C:\Users\DVORAKPA.COM\IMPEX\Desktop\Simplex Method(testovaci)\SimplexMethod\bin\Debug\SimplexMethod.EXE
2,30 1,90 1,70 1,70 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 83,00
1,30 1,30 2,20 2,70 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 97,00
1,50 2,30 2,60 1,60 0,00 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 91,00
1,20 2,00 1,30 2,70 0,00 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 86,00
1,50 1,60 2,20 2,40 0,00 0,00 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 86,00
2,30 1,90 1,70 1,70 4,30 3,90 3,80 3,20 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 129,00
1,30 1,30 2,20 2,70 3,60 3,10 3,70 3,40 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 134,00
1,50 2,30 2,60 1,60 4,30 4,10 4,50 3,60 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 154,00
1,20 2,00 1,30 2,70 4,30 4,50 4,30 3,60 0,00 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 123,00
1,50 1,60 2,20 2,40 3,20 3,20 4,30 4,10 0,00 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 140,00
59,90 47,40 59,70 52,90 11,30 16,40 11,60 12,50 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00
Z = 0
1,00 0,83 0,74 0,74 0,00 0,00 0,00 0,00 0,43 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 36,09
0,00 0,23 1,24 1,74 0,00 0,00 0,00 0,00 -0,57 1,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 50,09
0,00 1,06 1,49 0,49 0,00 0,00 0,00 0,00 -0,65 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 36,87
0,00 1,01 0,41 1,81 0,00 0,00 0,00 0,00 -0,52 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 42,70
0,00 0,36 1,09 1,29 0,00 0,00 0,00 0,00 -0,65 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 31,87
0,00 0,00 0,00 0,00 4,30 3,90 3,80 3,20 -1,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 46,00
0,00 0,23 1,24 1,74 3,60 3,10 3,70 3,40 -0,57 0,00 0,00 0,00 0,00 1,00 0,00 0,00 0,00 87,09
0,00 1,06 1,49 0,49 4,30 4,10 4,50 3,60 -0,65 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 99,87
0,00 1,01 0,41 1,81 4,30 4,50 4,30 3,60 -0,52 0,00 0,00 0,00 0,00 0,00 1,00 0,00 0,00 79,70
0,00 0,36 1,09 1,29 3,20 3,20 4,30 4,10 -0,65 0,00 0,00 0,00 0,00 0,00 0,00 0,00 1,00 85,87
0,00 -2,08 15,43 8,63 11,30 16,40 11,60 12,50 -26,04 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 -2161,61
Z = 2161,60869565217391
x_0 = 36,0869565217391
1,00 0,83 0,74 0,74 0,00 0,00 0,00 0,00 0,43 0,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 36,09
0,00 0,23 1,24 1,74 0,00 0,00 0,00 0,00 -0,57 1,00 0,00 0,00 0,00 0,00 0,00 0,00 0,00 50,09
0,00 1,06 1,49 0,49 0,00 0,00 0,00 0,00 -0,65 0,00 1,00 0,00 0,00 0,00 0,00 0,00 0,00 36,87
0,00 1,01 0,41 1,81 0,00 0,00 0,00 0,00 -0,52 0,00 0,00 1,00 0,00 0,00 0,00 0,00 0,00 42,70
0,00 0,36 1,09 1,29 0,00 0,00 0,00 0,00 -0,65 0,00 0,00 0,00 1,00 0,00 0,00 0,00 0,00 31,87
0,00 0,00 0,00 0,00 1,10 1,00 0,97 0,82 -0,26 0,00 0,00 0,00 0,00 0,26 0,00 0,00 0,00 11,79
0,00 0,23 1,24 1,74 0,18 0,00 0,68 0,86 0,23 0,00 0,00 0,00 -0,79 1,00 0,00 0,00 0,00 50,52
0,00 1,06 1,49 0,49 -0,22 0,00 0,51 0,24 0,40 0,00 0,00 0,00 0,00 -1,05 0,00 1,00 0,00 51,51
0,00 1,01 0,41 1,81 -0,66 0,00 -0,08 -0,09 0,63 0,00 0,00 0,00 0,00 -1,15 0,00 0,00 1,00 26,62
0,00 0,36 1,09 1,29 -0,33 0,00 1,18 1,47 0,17 0,00 0,00 0,00 0,00 -0,82 0,00 0,00 1,00 48,13
0,00 -2,08 15,43 8,63 -6,78 0,00 -4,38 -0,96 -21,84 0,00 0,00 0,00 0,00 -4,21 0,00 0,00 0,00 -2355,04
Z = 2355,04459308807
x_0 = 36,0869565217391
x_5 = 11,7948717948718
1,00 0,30 0,00 0,50 0,00 0,00 0,00 0,00 0,76 0,00 -0,50 0,00 0,00 0,00 0,00 0,00 0,00 17,81
0,00 -0,66 0,00 1,33 0,00 0,00 0,00 0,00 -0,02 1,00 -0,83 0,00 0,00 0,00 0,00 0,00 0,00 19,45
0,00 0,71 1,00 0,33 0,00 0,00 0,00 0,00 -0,44 0,00 0,67 0,00 0,00 0,00 0,00 0,00 0,00 24,72
0,00 0,71 0,00 1,68 0,00 0,00 0,00 0,00 -0,34 0,00 -0,28 1,00 0,00 0,00 0,00 0,00 0,00 32,48
0,00 -0,42 0,00 0,93 0,00 0,00 0,00 0,00 -0,17 0,00 -0,73 0,00 1,00 0,00 0,00 0,00 0,00 4,89
0,00 0,00 0,00 0,00 1,10 1,00 0,97 0,82 -0,26 0,00 0,00 0,00 0,00 0,26 0,00 0,00 0,00 11,79
0,00 -0,66 0,00 1,33 0,18 0,00 0,68 0,86 0,77 0,00 -0,83 0,00 0,00 -0,79 1,00 0,00 0,00 19,89
0,00 0,00 0,00 0,00 -0,22 0,00 0,51 0,24 1,05 0,00 -1,00 0,00 0,00 -1,05 0,00 1,00 0,00 14,64
0,00 0,71 0,00 1,68 -0,66 0,00 -0,08 -0,09 0,81 0,00 -0,28 0,00 0,00 -1,15 0,00 0,00 1,00 16,41
0,00 -0,42 0,00 0,93 -0,33 0,00 1,18 1,47 0,65 0,00 -0,73 0,00 0,00 -0,82 0,00
```

Obr. 5.7) Řešení testovací úlohy pouze programem Simplexové metody.

6 ZÁVĚR

Pro řešení rozsáhlých a komplikovaných optimalizačních úloh na počítači může výraznou pozitivní roli sehrát dekompozice a distribuovaný přístup. Namísto jedné rozsáhlé úlohy lze tak řešit několik menších vzájemně spolupracujících úloh. Složitost optimalizačních úloh obvykle bývá horší než lineární, a tak dochází k významné úspoře výpočetního času. Další úspora může vzniknout s ohledem na možnou paralelizaci při vhodné dekompozici (a vhodné úloze). Protože se řešení optimalizačních úloh stále častěji používá pro podporu rozhodování v reálném čase, může tento přístup významně rozšířit aplikační oblast matematického programování.

V práci byla ověřena první modelová realizace DOP. Bohužel bylo nutné pracovat v simulovaném prostředí, protože některé stěžejní části (jádro systému a komunikační knihovna OTP), které nebyly součástí této práce, nejsou dosud připraveny pro praktické použití. Přesto se podařilo realizovat potřebné programové moduly, odladit je a připravit je v takové podobě, aby po dokončení jádra systému komunikačních funkcí bylo nutné v řešiči provést jen minimální změny. Za cenné považuji, že se v rámci této práce podařilo systematicky realizovat funkční část systému DOP, kterou bude možné v dalším vývoji DOP použít a na kterou bude možné navazovat.

7 SEZNAM POUŽITÝCH ZDROJŮ

- [1] Griva I., Nash S. G., Sofer A.: Linear and Nonlinear Optimization, Virginia, George Mason University, Fairfax, 2009, 742 s., ISBN 978-0-898716-61-0.
- [2] Klapka J., Dvořák J., Popela P.: Metody operačního výzkumu, druhé vydání, Brno, Vysoké učení technické v Brně, Nakladatelství VUTIUM, 2001, 165 s., ISBN 80-214-1839-7.
- [3] Roupec, J.: Distribuované optimalizační problémy, Brno, VUT Brno, 2016, Habilitační práce
- [4] Popela P.: An Object-Oriented Approach to Multistage Stochastic Programming: Models and Algorithms. Phd Thesis, MFF UK Praha, 1998.
- [5] Popela P., Matoušek R., Sklenář J.: The Formal Framework For DOP. In Proceedings of the 14th International Conference on Soft Computing MENDEL'2008. Brno, 2008.
- [6] Popela P., Sklenář J., Matoušek R., Žampachová E., and Roupec J.: Advances in the Formal Framework for DOP, Proceedings of 17th International Conference of Soft Computing, MENDEL 2011, pp.320-325, ISBN 978-80-214-4302-0, (2011), 15.06.2011-17.06.2011.
- [7] Popela P.: Stochastic Programming, Malta, University of Malta, Faculty of Science, 2004
- [8] Popela P., Sklenář J., Matoušek R., Roupec J., Mrázková E. (2012): Advanced Decomposition Techniques Applied to DOP, Mendel Journal series, Vol. 2012, No.1, pp.582-587.

8 SEZNAM ZKRATEK

LP – Lineární programování

DOP – Distribuované optimalizační programy

OTP – Optimization Transfer Protocol

9 PŘÍLOHY

CD

- DOP – Master
- Zdrojové kódy
 - *Master*
 - *SimplexMethod*