ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta stavební
Katedra mechaniky

Hledání globálních optim příkladů rozměrové optimalizace

Search for global optima of sizing optimization benchmarks

Diplomová práce

Studijní program: Stavební inženýrství
Studijní obor: Konstrukce pozemních staveb

Vedoucí práce: Ing. Matěj Lepš, Ph.D.

Adéla Pospíšilová

Praha 2011

Zde je prostor pro zadání.

Zde je prostor pro rozšířené zadání.

Čestné prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pouze za odborného vedení vedoucího diplomové práce Ing. Matěje Lepše, Ph.D.

Dále prohlašuji, že veškeré podklady, ze kterých jsem čerpala, jsou uvedeny v seznamu použité literatury.

Datum: Podpis:

Na tomto místě bych chtěla srdečně poděkovat Ing. Matějovi Lepšovi, Ph.D. za inspiraci, jeho podporu, trpělivost, ochotu a mnoho cenných rad při vedení nejen mé diplomové práce, ale i v průběhu celého studia.

Dále bych chtěla poděkovat Ing. Anně Kučerové, Ph.D. a doc. Ing. Jaroslavu Kruisovi, Ph.D. za jejich cenné rady a připomínky.

Tato práce vznikla za podpory projektu č. SGS11/021/OHK1/1T/11 (Pokročilé algoritmy pro numerickou analýzu a modelování).

Abstrakt

Tato práce se zabývá hledáním globálních optim na klasických příkladech rozměrové diskrétní optimalizace. Pro menší konstrukce a analýzu okolí publikovaných lokálních optim je použita metoda hrubé síly. Pro větší konstrukce je potřeba použít efektivnější optimalizační metodu, a to metodu založenou na principu větví a mezí. Jsou-li správně nastavené hodnoty dolní a horní meze, prohledávaný prostor se značně omezí ovšem bez ztráty možnosti nalezení globálního optima.

K vyhodnocení omezujících podmínek optimalizační úlohy je třeba spočítat neznámé veličiny na konstrukci. Tento výpočet bude proveden mnohokrát. Proto je provedena důkladná analýza implementace metody konečných prvků a také porovnání řešičů soustav lineárních rovnic k získání co nejrychlejší rutiny pro vyhodnocování konstrukcí se stejnou topologií, ale měnícími se tuhostmi prutů.

Výpočet globálního optima na testovacích konstrukcích je výpočetně velmi náročný. Proto je potřeba využít paralelní výpočet jednak v rámci jednoho počítače pro otestování škálovatelnosti algoritmu a jednak v rámci zapojení počítačů do výpočetního clusteru.

Získaná optima ať už globální v případě 25-prutové konstrukce nebo lokální v rámci ostatních větších konstrukcí je třeba porovnat s optimy již publikovanými. Proto je v rámci této práce provedena důkladná rešerše literatury jednak z hlediska různých mutací zadání konstrukcí a jednak z hlediska publikovaných výsledků v rámci jedné nejčastěji používané mutace.

Abstract

This thesis focuses on searching for global optima of sizing optimization benchmarks. Enumeration was used for smaller structures and for analysis in the vicinity of published local optima. For larger structures it was necessary to use a more efficient optimization method based on branch and bound principles. If good lower and upper bounds are specified then searched space can be reduced still ensuring to find a global optima.

Unknown values of structures such as displacements and stresses were necessary to compute for the evaluation of constraints in an optimization problem. This computation will be performed many times. It was therefore necessary to carry out a careful implementation analysis of finite element method just as solvers for a system of linear equations. The goal was to find an efficient routine for evaluating structures with the same topology but different stiffness of rods.

Computational demands for obtaining global optima on benchmarks are very large. A suitable efficient parallelization was therefore necessary to apply to one multicore computer for scale testing or to a computer cluster for computing global optima.

An obtained global optimum for 25-bar structure or local optima for other larger structures were compared with published optima in available literature. Careful literature research was therefore realized for different mutations of optimization problems and optima for frequently used mutations were summarized.

Klíčová slova

tradiční příklady rozměrové optimalizace, diskrétní rozměrová optimalizace, metoda větví a mezí, globální optima, paralelní výpočet

Keywords

benchmarks, discrete sizing optimization, branch and bound method, global optima, paralel programming

Obsah

Seznam tabulek

Seznam obrázků

Kapitola 1

Úvod

V dnešní době je numerická optimalizace moderní a populární nástroj k získání jiného náhledu na konstrukce a jejich chování. Optimalizace se dá využít k nalezení vhodného tvaru konstrukce nebo jejího detailu například v podobě styčníků, dále je možné optimalizovat velikost průřezů, množství výztuže, tloušťku plechů, množství a pozice šroubů ve styku nebo třeba složení betonové směsi či kompozitu. Pro tyto reálné optimalizační úlohy je vyvíjen nespočet výpočetních metod, proto je dobré používat k jejich testování stejné typy úloh, aby se zjistila efektivnost a jejich chování. Testovací úlohy je dobré mít prozkoumané a znát jejich globální optima, tedy hodnoty účelové funkce včetně hodnot návrhových proměnných. V počátcích numerické optimalizace konstrukcí byl výpočetní výkon velmi slabý a nebylo tedy možné globální optima získat. Počítačový výkon však každým rokem roste a proto nastává vhodná doba pro analýzu těchto testovacích konstrukcí.

Existuje mnoho druhů optimalizačních metod pro získání optim úlohy jako jsou gradientní metody [68], heuristické metody [16] nebo evoluční algoritmy [17]. Tyto metody však nezaručují získání globálního optima. Dbá se u nich především na krátkou dobu výpočtu a nalezení alespoň optima lokálního případně nějakého dostatečně dobrého řešení. Neprohledává se totiž celý prostor řešení ale pouze jeho část, tím pádem mohou být dobrá řešení vynechána. Pro zaručené získání globálního optima je však škála optimalizačních nástrojů menší. Je možné použít metodu hrubé síly, kde se počítají všechna možná řešení, nebo metodu větví a mezí, kdy se omezí prostor pomocí horní a dolní meze, ale globální optimum zůstane v tomto prostoru zachováno.

Nárůst počítačového výkonu v rámci jednoprocesorového jednojádrového počítače již v poslední době není tak markantní [55]. Naproti tomu se vyrábí vícejadrové procesory a výkonné grafické karty. Počítače se sdružují do sítí, tzv. clusterů, a pomocí vhodných nástrojů je možné využít jejich výkon hromadně. Do sítě nemusí být zapojeny super výkonné stroje, proto je cluster relativně dostupnou technologií pro získání vyšší výpočetní síly. Využití vícejádrového procesoru v rámci jednoho počítače nebo výpočetního clusteru dnes umožňuje i komerční prostředí MATLAB [77] nebo veřejně

dostupné komunikační protokoly typu MPI (*Message Passing Interface*) [56]. Komunikace mezi jednotlivými procesy a rozdělení výpočetní úlohy mezi ně však není snadnou záležitostí a je třeba podrobit paralelní výpočet důkladné analýze, neboť ne každá úloha je pro paralelní způsob výpočtu vhodná.

Cílem této práce je najít optima na konstrukcích často zmiňovaných v souvislosti s rozměrovou optimalizací popsaných v kapitole 2, ve které je zároveň zavedena i konstrukce menší pro vývoj algoritmu metody větví a mezí. V kapitole 3 je uvedena metodika výpočtu neznámých na konstrukci, která je nezbytná pro vyhodnocení omezujících podmínek úloh, a její následná optimalizace. V kapitole 4 jsou popsány druhy metod pro získání optima diskrétní a spojité rozměrové optimalizace. V kapitole 5 jsou popsány způsoby paralelního výpočtu pro metodu větví a mezí jednak v prostředí MATLAB s využitím *Parallel Computing Toolbox*u a jednak v jazyce C/C++ s využitím protokolu MPI. V kapitole 6 jsou porovnána optima publikovaná v literatuře ke konstrukcím popsaným v kapitole 2 a optima získaná v rámci této práce.

Kapitola 2

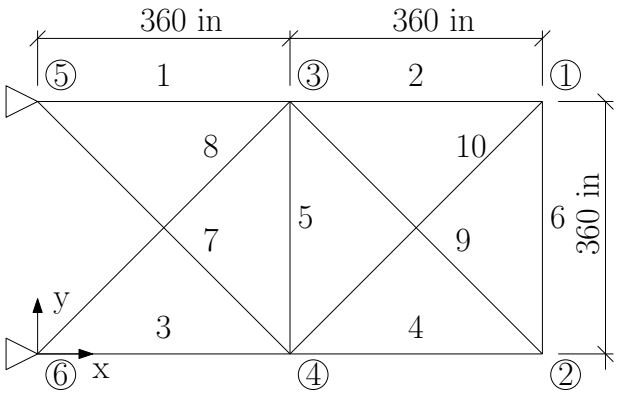
Testovací konstrukce

Testovací konstrukce (*Benchmarks*), které byly zvoleny pro tuto práci, patří k jedněm z nejčastěji používaných při rozměrové optimalizaci. Využívají se však i pro topologickou [28] či tvarovou optimalizaci [4]. Úlohy bývají jak v diskrétní, tak ve spojité verzi (vyjma 52-prutové konstrukce). Na první pohled se může zdát, že jsou ve všech publikovaných článcích naprosto totožné úlohy. Ty se nicméně vyskytují v různých mutacích, které v podstatě změní celou optimalizační úlohu. Výsledky pak již nejsou porovnatelné. I zkušení autoři leckdy tyto mutace přehlížejí a porovnávají neporovnatelné. Například autoři Rajeev a Krishnamoorthy v článku [64] porovnávají své výsledky pro 25-prutovou konstrukci s článkem [57], kde je však zavedeno o jeden zatěžovací stav více a je zde uvažováno omezení tlaku na vzpěr. Hmotnost konstrukce pak musí být logicky větší než při zavedení prutu pouze s prostým tlakem tak, jak jsou použity v [64], neboť jsou omezující podmínky přísnější. Autoři jako např. Kripka v [43] pak tyto hodnoty přebírají bez kontroly a vznikají opakující se omyly.

2.1 Desetiprutová příhradová 2D konzola

Tato konstrukce byla nejspíše prvně použita jako **spojitá úloha** v článku publikovaným autorem Venkayyou roku 1971 [83]. Od té doby se její topologie stále používá, ať už v podobě původní (viz obrázek č. 2.1) nebo modifikované. U modifikované verze mohou být rozměry v SI jednotkách, ale ekvivalentní k původní verzi v jednotkách anglosaských [59], [37], anebo vůbec v odlišných rozměrech [35].

První výskyt **diskrétní úlohy** se stejnou topologií jako na obrázku č. 2.1 byl nejspíše v roce 1980 v článku autorů Fleuryho a Schmita [21]. Profily pro jednotlivé pruty jsou ale vybírány z mnohem menší sady než se používá v dnešní době. Výskyt zadání úlohy pro diskrétní optimalizaci, která se dnes používá asi nejčastěji a která bude užita i v této práci, je nejspíše v článku autorů Rajeeva a Krishnamoorthyho z roku 1992 [64]. Sada profilů je zde zadána tak, jako v tabulce 2.1 pod písmenem A. Ostatní sady, které se v literatuře také vyskytují, lze nalézt v téže tabulce.



Obrázek 2.1: 10-prutová příhradová 2D konzola

A[a]	1,62; 1,80; 1,99; 2,13; 2,38; 2,62; 2,63; 2,88; 2,93; 3,09; 3,13; 3,38; 3,47; 3,55; 3,63; 3,84; 3,87; 3,88; 4,18; 4,22; 4,49; 4,59; 4,80; 4,97; 5,12; 5,74; 7,22; 7,97; 11,50; 13,50; 13,90; 14,20; 15,50; 16,00; 16,90; 18,80; 19,90; 22,00; 22,90; 26,50; 30,00; 33,50
B[b]	0,1; 0,5; 1; 1,5; 2; 2,5; 3; 3,5; 4; 4,5; 5; 5,5; 6; 6,5; 7; 7,5; 8; 8,5; 9; 9,5; 10; 10,5; 11; 11,5; 12; 12,5; 13; 13,5; 14; 14,5; 15; 15,5; 16; 16,5; 17; 17,5; 18; 18,5; 19; 19,5; 20; 20,5; 21; 21,5; 22; 22,5; 23; 23,5; 24; 24,5; 25; 25,5; 26; 26,5; 27; 27,5; 28; 28,5; 29; 29,5; 30; 30,5; 31; 31,5
C[c]	(0,1); 0,347; 0,440; 0,539; 0,954; 1,081; 1,174; 1,333; 1,488; 1,764; 2,142; 2,697; 2,8; 3,131; 3,565; 3,813; 4,805; 5,952; 6,572; 7,192; 8,525; 9,3; 10,850; 13,330; 14,290; 17,170; 19,180; 23,680; 28,080; 33,7
D[d]	0,1; 3,9379; 5,569; 5,7447; 7,9379; 8,0621
E[d]	0,1; 1,0; 2,0; 3,0; 4,0; 5,0; 6,0; 7,0; 8,0; 9,0
F[d]	0,1; 0,5565; 7,4683; 15,286; 21,198; 21,618; 23,274; 30,031
G[d]	0,1; 1,0; 2,0; 3,0; 4,0; 5,0; 6,0; 7,0; 8,0; 9,0; 10,0; 12,0; 14,0; 16,0; 18,0; 20,0; 22,0; 24,0; 26,0; 28,0; 30,0; 32,0; 34,0; 36,0

[a] [10], [13], [14], [24], [27], [43], [50], [53], [58], [64], [69], [82], [88]
[b] [53], [88]
[c] [11]
[d] [63]

Tabulka 2.1: Sady používané v literatuře pro 10-prutovou konstrukci v in^2

Úloha pro spojitou optimalizaci mívá zadanou dolní a horní hranici ploch příčného řezu. Velice často se používá dolní hranice 0,1 in^2 [50], méně často pak 0,01 in^2 [74]. Horní hranice v této mutaci nebývá definována nebo využívá hodnoty 10 in^2 jako např.

v článku [12]. V souvislosti s diskrétní úlohou se sadou A z tabulky 2.1 vznikla i mutace, která kopíruje nejmenší plochu ze sady, tedy 1,62 in^2 jako dolní hranici a největší plochu 33,5 in^2 jako horní hranici [69]. V této práci se využívá právě tohoto zadání.

Statické **zatížení konstrukce** bývá nejčastěji svislé na styčnících 2 a 4 znázorněných na obrázku 2.1 v hodnotě 100 kipů. Toto zatížení lze nalézt jak u spojité [83], tak u diskrétní úlohy [64]. Existují však i úlohy se zatěžovacími stavy s jednou [25] či více [54] vodorovnými silami anebo svislými silami i na ostatních volných styčnících [83]. Souhrn statických zatěžovacích stavů užívaných v literatuře lze nalézt v tabulce č. 2.2. Existují však i úlohy s dynamickým zatížením, které využívají buď harmonickou budící sílu [82] anebo ekvivalentní statické zatížení [38].

Zatížení č.	Styčník[a]	Směr zatížení		jednotky	reference
		y	x		
1	2	-100	0	kipy	[5], [6], [10], [11], [12], [13], [14], [20], [21], [24], [25], [28], [31], [32], [41], [43], [47], [48], [50], [51], [52], [53], [58], [60], [63], [65], [67], [69], [74], [82], [83], [88], [91]
	4	-100	0		
2	2	-667 340	0	N	[37]
	4	-667 340	0		
	1	222 450	0		
	3	222 450	0		
3	2	-444,8	0	N	[59]
	4	-444,8	0		
	1	0	44,5		
	2	0	44,5		
4	2	-100	0	kipy	[25], [54]
	4	-100	0		
	2	0	400		
5	2	-150	0	kipy	[41], [47], [48], [52], [67], [74]
	4	-150	0		
	1	50	0		
	3	50	0		

[a] (čísla styčníků odpovídají obrázku 2.1)

Tabulka 2.2: Zatížení používaná pro 10-prutovou konstrukci

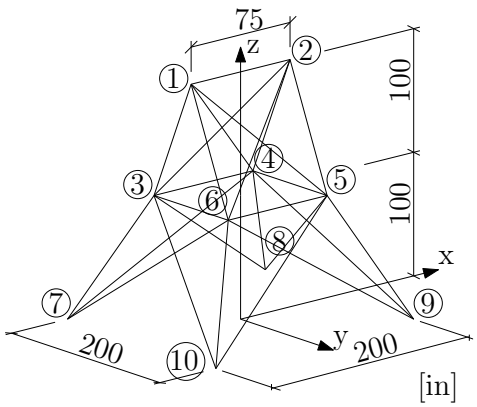
Úloha je zadávána včetně **omezujících podmínek**. Nejčastěji bývá maximální posun omezen ±2 in a maximální napětí ±25 ksi [83]. Tyto podmínky se mohou u spojité optimalizace vyskytovat buď samostatně [83], nejčastěji formou omezení napětí

(1. mezní stav), anebo dohromady [50], [83]. Obou podmínek najednou bývá užíváno u diskrétního zadání úlohy. Pro spojitou úlohu se pak vyskytují i zadání s omezením napětí na 9. prutu dle obrázku č. 2.1 ±75 ksi [27].

Užitý **materiál** bývá hliník s objemovou hmotností 0,1 lb/in^3 a modulem pružnosti 10^4 ksi [83]. Lze však nalézt i úlohy s blíže nepopsaným materiálem s modulem pružnosti 29 000 ksi [35], 30 000 ksi [51] anebo s použitím oceli [59].

V této práci bude užita 10-prutová konstrukce se zatížením č. 1 z tabulky 2.2, s omezením napětí na všech prutech ±25 ksi a všech posunů ve vodorovném i svislém směru ±2 in. Materiálem je hliník s objemovou hmotností 0,1 lb/in^3 a modulem pružnosti 10^4 ksi. U diskrétní úlohy bude uvažována sada profilů A z tabulky 2.1 a u spojité úlohy bude dolní mez 1,62 in^2 a horní mez 33,5 in^2.

2.2 Dvacetipětiprutová příhradová 3D věž



Obrázek 2.2: 25-prutová příhradová 3D věž

Tato konstrukce byla prvně zveřejněna autory Foxem a Schmitem roku 1966 v článku [23], kde se jednalo o spojitou úlohu. Její **topologie** se, až na sporadické výjimky (viz [39]), kde je použit převod do SI jednotek, dodnes používá v nezměněné podobě (viz obrázek č. 2.2). Jelikož je konstrukce symetrická podle osy xz a yz, je možné pruty rozdělit do skupin viz tabulka 2.3 a tím snížit výpočetní náročnost. Toto rozdělení použili již Fox a Schmit a je používáno dodnes. **Diskrétní úloha** pro tuto konstrukci byla poprvé publikována rok poté autory Gellatlym a Marcalem v [26]. Sada profilů byla vytvořena pomocí přírůstků ploch 0,4 in^2 nebo 0,8 in^2. Sada, která je v dnešní době velice často používána, byla v souvislosti s touto konstrukcí uveřejněna v článku autorů Rajeeva a Krishnamoorthyho [64] a lze ji nalézt v tabulce 2.4 pod písmenem A. V téže tabulce a tabulce č. 2.7 jsou pak i ostatní využívané sady profilů.

Skupina	Připojení prutů k uzlům	Omezení napětí		
		Tah	Tlak - var. 1	Tlak - var. 2
		psi	psi	psi
A_1	1-2	40 000	-40 000	-35 092
A_2	1-4, 2-3, 1-5, 2-6	”	”	-11 590
A_3	2-5, 2-4, 1-3, 1-6	”	”	-17 305
A_4	3-6, 4-5	”	”	-35 092
A_5	2-4, 5-6	”	”	-35 092
A_6	3-10, 6-7, 4-9, 5-8	”	”	-6 759
A_7	3-8, 4-7, 6-9, 5-10	”	”	-6 959
A_8	3-7, 4-8, 5-9, 6-10	40 000	-40 000	-11 082

Tabulka 2.3: Rozdělení prutů do skupin pro 25-prutovou konstrukci

A[a]	0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1,0; 1,1; 1,2; 1,3; 1,4; 1,5; 1,6; 1,7; 1,8; 1,9; 2,0; 2,1; 2,2; 2,3; 2,4; 2,5; 2,6; 2,8; 3,0; 3,2; 3,4
B[b]	0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; 1,0; 1,1; 1,2; 1,3; 1,4; 1,5; 1,6; 1,7; 1,7; 1,8; 1,9; 2,0; 2,1; 2,2; 2,3; 2,4; 2,5; 2,6; 2,7; 2,8; 2,9; 3,0; 3,1; 3,2; 3,3; 3,4; 3,5
C[c]	0,01; 0,4; 0,8; 1,2; 1,6; 2; 2,4; 2,8; 3,2; 3,6; 4; 4,4; 4,8; 5,2; 5,6; 6
D[d]	0,01; 0,1; 0,2; 0,3; 0,4; 0,5; 0,6; 0,7; 0,8; 0,9; . . . ; 5,4; 5,5; 5,6
E[d]	0,01; 0,8; 1,6; 2,4; 3,2; 4; 4,8; 5,6
F[e]	0,01; 0,6853; 1,6217; 2,0476; 2,6712; 2,9965

[a] [13], [18], [43], [49], [50], [53], [57], [58], [63], [64], [87], [88]

[b] [82]

[c] [40], [49], [53], [88]

[d] [21]

[e] [63]

Tabulka 2.4: Sady používané v literatuře pro 25-prutovou konstrukci v in^2

Aby plochy nenabývaly nekonečně malých hodnot, bývá ve **spojité optimalizaci** nastavována dolní mez na hodnotu 0,01 in^2 [91]. Jelikož však bude využita spojitá optimalizace pro potřeby optimalizace diskrétní, je třeba, aby byly počáteční hodnoty konzistentní. Proto bude v této práci nastavena dolní mez na plochu 0,1 in^2, což je nejmenší profil ze sady A tabulky 2.4, a horní mez na plochu 3,4 in^2, což je největší profil z téže sady.

Zatížení	Zatěžovací stav	Uzel	F_x	F_y	F_z	jednotky
A[a]		1	1	-10	-10	kips
		2	0	-10	-10	
		3	0.5	0	0	
		6	0.6	0	0	
B[b]	1	1	0	20	-5	kips
		2	0	-20	-5	
	4	1	1	10	-5	
		2	0	10	-5	
		3	0.5	0	0	
		6	0.5	0	0	
C[c]		1	4 453.74	-44 537.4	-44 537.4	N
		2	0	-44 537.4	-44 537.4	
		3	2 226.87	0	0	
		6	2 672.24	0	0	
D[d]	1	1	1	10	-5	kips
		2	0	10	-5	
		3	0.5	0	0	
		6	0.6	0	0	
	2	1	0	20	-5	
		2	0	-20	-5	
E[e]	1	1	1	10	-5	kips
		2	0.5	0	0	
	2	1	0	20	-5	
		2	0	-20	-5	

[a] [18], [50], [58], [60], [82], [88]
[b] [1], [21], [23], [44], [47], [48], [52], [53], [63], [65], [67], [74], [83], [84], [91]
[c] [43], [64]
[d] [57]
[e] [26]

Tabulka 2.5: Zatížení používané v literatuře pro 25-prutovou konstrukci

Zatížení, které se pro konstrukci v literatuře používá, lze nalézt v tabulce 2.5. Pod písmenem B je uvedeno původní zatížení se dvěma zatěžovacími stavy, které se pro spojitou optimalizaci používá dodnes. V roce 1986 vznikla jeho mutace, v tabulce uvedena pod písmenem D, která mění x-ovou složku zatížení na uzlu 6 v zatěžovacím stavu 1. V roce 1992 pak autoři Rajeev a Krishnamoorthy používají pouze 1. zatěžovací stav tohoto zatížení v SI jednotkách (v tabulce pod písmenem C) a v roce 1995 vznikla mutace, která se pro diskrétní optimalizaci ustálila. V tabulce je uvedena pod písmenem A. Tohoto zatížení bude využito i v této práci.

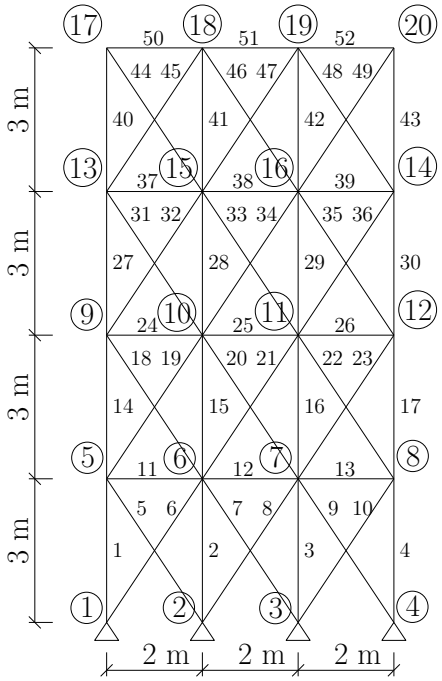
Úloha je podmíněnou optimalizací a jsou pro ni tedy zadány **omezující podmínky**, většinou ve formě omezení maximální hodnoty napětí a maximální hodnoty posunu. Absolutní hodnota tahu a tlaku se buď zadává stejná (viz sloupce Tah a Tlak - varianta 1 tabulky 2.3) [64], a to 40 ksi, anebo je kladen důraz i na vzpěr prutů a jednotlivé

hodnoty napětí v tlaku jsou zadány pro každou skupinu zvlášť [23]. Hodnoty lze nalézt ve sloupci Tlak - varianta 2 v téže tabulce. Omezení posunů bývá pro vodorovný i svislý směr $\pm 0,35$ in.

Materiálem, který se pro tuto konstrukci používá, je hliník s objemovou hmotností 0,1 lb/in^3 a modulem pružnosti 10^4 ksi [23].

Pro účely **optimalizace v této práci** je použita konstrukce s topologií na obrázku 2.2 s rozdělením prutů do skupin dle tabulky 2.3. Je použito zatížení A z tabulky 2.5. Omezující podmínky jsou dány jako omezení posunu ve vodorovném i svislém směru $\pm 0,35$ in a omezením napětí ± 40 ksi, které je také zapsáno v tabulce 2.3 ve sloupcích tah a tlak - varianta 1. Pro diskrétní úlohu je vybíráno ze sady profilů A z tabulky 2.4 a pro spojitou úlohu, která bude zapotřebí jako vstup úlohy diskrétní, je použita jako dolní mez plocha 0,1 in^2 a jako horní mez 3,4 in^2.

2.3 Padesátidvouprutová konstrukce



Obrázek 2.3: 52-prutová příhradová 2D konstrukce

	Pruty
A_1	1, 2, 3, 4
A_2	5, 6, 7, 8, 9, 10
A_3	11, 12, 13
A_4	14, 15, 16, 17
A_5	18, 19, 20, 21, 22, 23
A_6	24, 25, 26
A_7	27, 28, 29, 30
A_8	31, 32, 33, 34, 35, 36
A_9	37, 38, 39
A_{10}	40, 41, 42, 43
A_{11}	44, 45, 46, 47, 48, 49
A_{12}	50, 51, 52

Tabulka 2.6: Sdružení ploch příčných průřezů do skupin (52-prutová konstrukce)

A[a]	0,111; 0,141; 0,196; 0,250; 0,307; 0,391; 0,442; 0,563; 0,602; 0,766; 0,785; 0,994; 1; 1,228; 1,266; 1,457; 1,563; 1,62; 1,8; 1,99; 2,13; 2,38; 2,62; 2,630; 2,88; 2,93; 3,09; 3,13; 3,38; 3,47; 3,55; 3,63; 3,84; 3,87; 3,88; 4,18; 4,22; 4,49; 4,59; 4,8; 4,97; 5,12; 5,74; 7,22; 7,97; 8,53; 9,3; 10,85; 11,5; 13,5; 13,9; 14,2; 15,5; 16; 16,9; 18,8; 19,9; 22; 22,9; 24,5; 26,5; 28; 30; 33,5
B[b]	71,613; 90,968; 126,451; 161,29; 198,064; 252,258; 285,161; 363,225; 388,386; 494,193; 506,451; 641,289; 645,16; 792,256; 816,773; 940; 1008,385; 1045,159; 1161,288; 1283,868; 1374,191; 1535,481; 1690,319; 1696,771; 1858,061; 1890,319; 1993,544; 2019,351; 2180,641; 2238,705; 2290,318; 2341,191; 2477,414; 2496,769; 2503,221; 2696,769; 2722,575; 2896,768; 2961,284; 3096,768; 3206,445; 3303,219; 3703,218; 4658,055; 5141,925; 5503,215; 5999,998; 6999,986; 7419,34; 8709,66; 8967,724; 9161,272; 9999,98; 10322,56; 10903,204; 12129,008; 12838,684; 14193,52; 14774,164; 15806,42; 17096,74; 18064,48; 19354,8; 21612,86

[a] v in^2 [40], [49], [53], [88]

[b] v mm^2 [28], [40], [50], [53], [88]

Tabulka 2.7: Sada ASIC používaná v literatuře pro 25-prutovou a 52-prutovou konstrukci

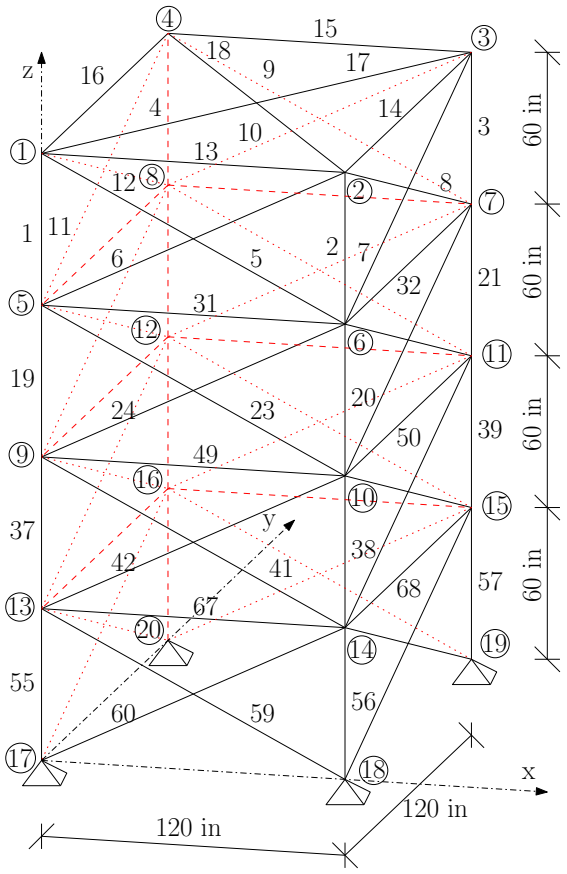
Tato konstrukce je ze všech zde uvedených nejmladší. Poprvé byla publikována v článku autorů Wu a Chowa [88] v roce 1995. Vyskytuje se pouze v **diskrétní verzi** v původním zadání většinou v SI jednotkách, méně v anglosaských [49]. Konstrukce je symetrická, proto je možné jednotlivé pruty opět rozdělit do skupin tak, jak je uvedeno v tabulce 2.6. Profily pro skupiny se vybírají ze sady v tabulce 2.7 v mm^2. Pro konstrukci s rozměry zadanými v anglosaských jednotkách je v téže tabulce i sada s plochami profilů v in^2.

Zatížení konstrukce se uvádí jako síly ve vodorovném směru F_x o velikosti 100 kN a ve svislém směru F_y o velikosti 200 kN na všech uzlech horního patra tedy 17 až 20. Pro úlohu v anglosaských jednotkách je síla ve směru x 22,48 kipů a síla ve směru y 44,9 kipů na uzlech 17 až 20 [49].

Omezující podmínkou je v této úloze pouze omezení napětí v tahu i tlaku ± 180 MPa respektive $\pm 26,01$ ksi. Použitým **materiálem** je ocel s objemovou hmotností 7860 kg/m^3 a modulem pružnosti $2,07 \cdot 10^5$ MPa. Pro anglosaské jednotky je objemová hmotnost materiálu 0,284 lb/in^3 a modul pružnosti 30 022,8 ksi.

Konstrukce, která bude užita **v této práci**, přebírá všechny hodnoty v SI jednotkách, neboť jsou tyto výsledky lépe porovnatelné s publikovanými.

2.4 Sedmdesátidvouprutová konstrukce



Obrázek 2.4: 72-prutová příhradová 3D konstrukce

	Pruty
A_1	1, 2, 3, 4
A_2	5, 6, 7, 8, 9, 10, 11, 12
A_3	13, 14, 15, 16
A_4	17, 18
A_5	19, 20, 21, 22
A_6	23, 24, 25, 26, 27, 28, 29, 30
A_7	31, 32, 33, 34
A_8	35, 36
A_9	37, 38, 39, 40
A_{10}	41, 42, 43, 44, 45, 46, 47, 48
A_{11}	49, 50, 51, 52
A_{12}	53, 54
A_{13}	55, 56, 57, 58
A_{14}	59, 60, 61, 62, 63, 64, 65, 66
A_{15}	67, 68, 69, 70
A_{16}	71, 72

Tabulka 2.8: Sdružení ploch příčných průřezů do skupin (72-prutová konstrukce)

Topologie této konstrukce byla poprvé publikována v roce 1968 v technické zprávě autorem Venkayyou a kol. [84] pro případ spojité optimalizace a v nezměněné podobě je používána dodnes. Jediná modifikace, která se v literatuře vyskytuje, je převod do SI jednotek se zachováním stejných rozměrů [66]. Konstrukce je symetrická, proto je možné rozdělit pruty do skupin tak, jak je uvedeno v tabulce 2.8. Toto dělení se začalo používat od roku 1980 v článku autorů Fleuryho a Schmita [21]. **Diskrétní úloha** s touto konstrukcí byla publikována v roce 1995 v článku autorů Wu a Chowa [88]. Profily je možné vybírat buď ze sady uvedené v tabulce 2.7 v anglosaských jednotkách anebo ze sady: 0,1; 0,2; 0,3; ...; 3,2 in^2 pokaždé s inkrementem 0,1 in^2 [53], [88], [40]. U **spojité úlohy** je dolní mez plochy profilu nastavena buď na hodnotu 0,1 in^2 [84], což je častější případ, anebo na 0,01 in^2 [48]. Horní mez nebývá nikterak omezena.

Původní **zatížení** z článku [84] je v tabulce 2.9 pod písmenem A. Jelikož pruty v této úloze ještě nebyly rozděleny do skupin a autoři nejspíše zamýšleli navrhnout symetrickou konstrukci, byl zatěžovací stav č. 1 postupně otočen o 90° do všech styčníků horního patra a tím vznikly zatěžovací stavy s číslem 2 až 4. V roce 1980 autoři

Zatížení	Zatěžovací stav	Uzel	F_x	F_y	F_z	reference
A	1	1	5	5	-5	[84]
	2	2	-5	5	-5	
	3	3	-5	-5	-5	
	4	4	5	-5	-5	
	5	1	0	0	-5	
		2	0	0	-5	
		3	0	0	-5	
		4	0	0	-5	
B	1	1	5	5	-5	[1]
	2	1	0	0	-5	
		2	0	0	-5	
		3	0	0	-5	
		4	0	0	-5	
C		1	5	5	-5	[33]
		2	0	0	-5	
		3	0	0	-5	
		4	0	0	-5	

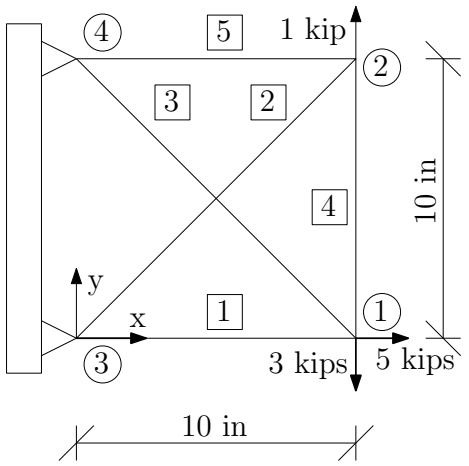
Tabulka 2.9: Zatížení používané v literatuře pro 72-prutovou konstrukci v kipech

Fleury a Schmit uvádějí rozdělení do skupin a tím pádem odpadá nutnost přepočítávat omezující podmínky pro 2. až 4. zatěžovací stav ze zatížení A. Vzniklé zatížení je uvedeno v téže tabulce pod písmenem B. V tabulce je také uvedena další modifikace pod písmenem C, která se nicméně neuchytila.

Úloha má **omezující podmínky** ve formě omezení maximální hodnoty napětí ± 25 ksi a maximální hodnoty posunu uzlů ve vodorovném i svislém směru $\pm 0,25$ in. Někdy se vyskytuje samostatně pouze 1. mezní stav, tedy omezení napětí [33], většinou jsou však podmínky omezení maximálního napětí a maximálního posunu uvedeny společně [84]. Použitým **materiálem** je opět hliník s objemovou hmotností 0,1 lb/in^3 a modulem pružnosti 10^4 ksi [84].

V této práci bude použita topologie s rozměry v anglosaských jednotkách pro lepší porovnání výsledků. Jelikož jsou pruty rozděleny do skupin, bude použito zatížení B z tabulky 2.9. Hodnota maximálního napětí je ± 25 ksi a hodnota maximální posunu $\pm 0,25$ in ve vodorovném i svislém směru. Materiálem je hliník. Pro diskrétní optimalizaci bude použita sada 0,1; 0,2; 0,3; ...; 3,2 in^2 s inkrementem 0,1 in^2 a pro spojitou optimalizaci bude nastavena dolní mez na 0,1 in^2 a horní mez na 3,2 in^2.

2.5 Pětiprutová příhradová konstrukce



Obrázek 2.5: Pětiprutová příhradová 2D konstrukce

Poslední konstrukce, která bude v práci využita, je pětiprutová příhradová konstrukce znázorněná na obrázku 2.5. Její topologie je uveřejněna v [46]. Tato konstrukce není v souvislosti s rozměrovou optimalizací zmiňována, nicméně pro potřeby testování algoritmů a jejich verifikaci bylo potřeba zavést dostatečně malou, ale zároveň reprezentativní konstrukci. Proto byla pro **diskrétní optimalizaci** vybrána sada profilů 0,01; 0,02; 0,03; ...; 0,1 in^2 s inkrementem 0,01 in^2. **Spojitá úloha** kopíruje nejmenší a největší profil ze sady pro dolní (0,01 in^2) a horní mez (0,1 in^2).

Zatížení je znázorněno na obrázku 2.5. **Omezující podmínky** jsou ve formě omezení napětí a posunů. Maximální napětí, které je v prutech povoleno, je ± 60 ksi a maximální povolená hodnota posunu styčníku ve vodorovném i svislém směru je $\pm 0,06$ in. Použitým **materiálem** je hliník s objemovou hmotností 0,1 lb/in^3 a modulem pružnosti 10^4 ksi.

Kapitola 3

Metodika a řešení rychlosti výpočtu neznámých na konstrukci

Při výpočtu omezujících podmínek optimalizační úlohy bude zapotřebí provést analýzu konstrukce, tedy spočítat neznámé posuny a napětí na konstrukci. Jelikož se tyto omezující podmínky budou vyhodnocovat velmi často, je vhodné provést optimalizaci kódu a zamyslet se nad případným rychlejším nebo více efektivním způsobem výpočtu, aby se co nejvíce uspořil výpočetní čas. Některé myšlenky z této kapitoly byly uvedeny již v bakalářské práci [61], nicméně je na místě uvést alespoň menší shrnutí včetně nově získaných poznatků.

3.1 Řešení rychlosti výpočtu metody konečných prvků

K výpočtu neznámých posunů a napětí byla zvolena deformační varianta metody konečných prvků (MKP), neboť je snadno implementovatelná a s daným kódem je možné dále pracovat. Stručné odvození této metody pro tažený resp. tlačený prut lze nalézt v příloze B, detailněji v [61] anebo v [19]. Implementace bude provedena jak v prostředí MATLAB, tak v jazyce C/C++.

3.1.1 Implementace MKP v Matlabu a její analýza

Algoritmus, který řeší výpočet neznámých posunů a napětí, by se dal popsat v následujících bodech.

1. Nejprve je třeba zadat inicializační hodnoty parametrů konstrukce, a to například materiálové vlastnosti, topologii konstrukce, informace o kódových číslech, zatížení nebo podepření konstrukce.

2. Poté se sestaví matice kódových čísel.

3. Proběhne postupné sestavení lokálních matic tuhosti prutů a provede se lokalizace, tedy umístění lokální matice tuhosti prutů do globální matice konstrukce.

4. Vyřeší se soustava lineárních rovnic $K \cdot r = f$ k získání neznámých posunů r, přičemž K je matice tuhosti konstrukce a f je vektor uzlových zatížení.

5. Spočítají se vnitřní síly.

6. Vyhodnotí se absolutní hodnota maximálního napětí a absolutní hodnota maximálního posunu ve vodorovném a svislém směru, což jsou omezující podmínky pro většinu konstrukcí (vyjma 52-prutové, u té se vyhodnocuje pouze napětí) popsaných v kapitole 2, a spočítá se hmotnost konstrukce.

Sestavený kód je vhodné analyzovat pomocí nějakého dostupného nástroje a zjistit, které části je třeba sestavit efektivněji. Program MATLAB takovéto nástroje nabízí například ve formě *Profiler*u. *Profiler* se zavolá na začátku kódu pomocí příkazu `profile on`. Na konci části analyzovaného kódu je *Profiler* ukončen příkazem `profile off`. Okno s výsledky analýzy se zobrazí příkazem `profile viewer`.

výpis kódu	počet volání	čas v s	čas v %
`Disp=s\f;`	1	0,006	40,0%
`Stiff=zeros(2*numnod);`	1	0,005	33,3%
ostatní		0,004	26,7%
celkem		0,015	100%

Tabulka 3.1: Výstup z Profileru MATLABu pro jedno spuštění kódu přímého řešiče pro desetiprutovou konstrukci (kód je dostupný na přiloženém CD)

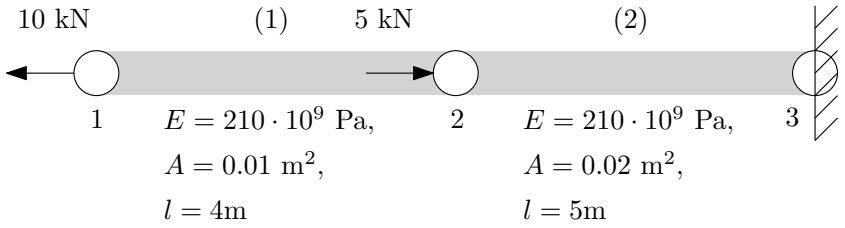
výpis kódu	počet volání	čas v s	čas v %
`k=Area(m)*Emod(m)/Length(m)*[T...`	10 000	0,065	15,2%
`ij=[2*inode'-1,2*inode',2*jnod...`	1 000	0,046	10,7%
`T=[[cc,cs];[cs,ss]];`	10 000	0,040	9,3%
`Stiff(n,n)=Stiff(n,n)_k;+`	10 000	0,038	8,9%
`f([2*idfx'-1;2*idfy'])=[fx';fy...`	1 000	0,029	6,8%
ostatní		0,210	49,1%
celkem		0,428	100%

Tabulka 3.2: Výstup z Profileru MATLABu pro 1 000 spuštění kódu přímého řešiče pro 10-prutovou konstrukci (kód je dostupný na přiloženém CD)

Profiler pro jedno spuštění kódu oznámí, že nejvíce výpočetního času je věnováno výpočtu posunů, tedy řešení soustavy lineárních rovnic, viz tabulka 3.1. Při spuštění kódu tisíckrát za sebou je nejvíce času věnováno sestavení globální matice tuhosti, viz tabulka 3.2. Je tedy potřeba zamyslet se nad alternativami implementace těchto bodů.

3.1.2 Vyjádření matice tuhosti konstrukce pomocí parametrů

Pro každé vyhodnocení sady omezujících podmínek bude zapotřebí konstrukci znovu přepočítat. Jelikož se jedná o rozměrovou optimalizaci, u které se nemění topologie konstrukce, ale pouze tuhosti jednotlivých prutů, není třeba matici tuhosti sestavovat při každém výpočtu. Stačí si ji vyjádřit v závislosti na zvoleném parametru, například na ploše příčného řezu prutu nebo na jeho tuhosti. Do této matice se poté v běhu skriptu pouze dosazuje a uspoří se tak výpočetní čas. Parametrické vyjádření se dá získat pomocí symbolického toolboxu (*Symbolic Math Toolbox*), kde se dopředu definují parametry jako symbolické proměnné, a to `syms k1 k2 ... kn real`, a poté se s nimi počítá jako s normálními reálnými proměnnými. Je ovšem třeba dát pozor na alokaci paměti vektorů a matic, například matice rozměrů `n x m` je alokována jako `Matrix=sym(zeros(n,m))`.



Obrázek 3.1: Dvouprutová příhradová konstrukce se zadanými okrajovými podmínkami

Pokud by sestavení matice tuhosti konstrukce probíhalo při každém běhu skriptu, výňatek pseudokódu a výsledná matice tuhosti by pro konstrukci na obrázku 3.1 vypadaly takto.

```
k = [525000, 840000] %kN/m
lokalizace
K
>>
K =
      525000     -525000           0
     -525000     1365000     -840000
           0     -840000      840000
```

Matice tuhosti konstrukce je vyjádřená číselně a tedy použitelná jen pro jedinečné zadání. Po parametrizaci daného kódu je ale matice pro tutéž topologii s měnícími se tuhostmi prutů dostatečně obecná.

```
syms k1 k2 real
k = [k1, k2]
lokalizace
K
>>
```

```
K =
       k1      -k1        0
      -k1    k1+k2      -k2
        0      -k2       k2
```

Matice tuhosti konstrukce s parametrem se pak po drobných syntaktických úpravách použije místo lokalizace v předchozím kódu a číselná matice tuhosti se získá pouze dosazením za parametry `k1` a `k2`.

Takovéto parametrické vyjádření by teoreticky šlo použít i pro vypočtené posuny. Jelikož je ale u výpočtu soustavy lineárních rovnic hodně operací, ani u 10-prutové konstrukce nebylo možné posuny v závislosti na parametru vypočítat. Výpočet vnitřních sil respektive napětí ovšem již tak výpočetně náročný není, proto se opět dají použít parametry mající význam neznámých posunů a jednotlivých tuhostí prutů. Výsledný algoritmus po úpravách pak může být shrnut v následujících bodech.

1. Zadají se veškeré inicializační parametry pro konstrukci jako v předchozím algoritmu v bodě 1.
2. Vypočítají se tuhosti jednotlivých prutů.
3. Tuhosti prutů se dosadí do parametricky vyjádřené matice tuhosti konstrukce a ta se vyčíslí.
4. Vypočítají se neznámé posuny ze soustavy lineárních rovnic.
5. Vypočtené posuny se dosadí do parametricky vyjádřeného vektoru vnitřních sil a ten se vyčíslí.
6. Vyhodnotí se absolutní hodnota maximálního napětí a absolutní hodnota maximálního posunu ve vodorovném a svislém směru a spočítá se hmotnost konstrukce.

Porovnání doby trvání výpočtu lze nalézt v podkapitole 3.3 pro testovací konstrukce popsané v kapitole 2.

3.1.3 Způsoby uložení matice tuhosti

Matici tuhosti konstrukce lze uložit do paměti mnoha způsoby. Nejjednodušším způsobem je klasická plná matice obsahující všechny prvky, tedy i ty nulové. Uložení plné matice však může být nevýhodné, obsahuje-li matice mnoho nulových prvků. Pak se nabízí uložení matice jako řídké. Řídká matice obsahuje pouze nenulové prvky a tím se snižují nároky na operační paměť. Tento způsob ovšem není samospasitelný a automaticky použitelný pro všechny matice tuhosti, které v drtivé většině alespoň nějaké nulové prvky obsahují. K řídké matici se přistupuje odlišně, prvek po prvku,

a pro některé matice, které naopak obsahují pouze málo nulových prvků, může výpočet trvat déle.

Prakticky každá matice tuhosti konstrukce má pásový charakter. Pokud se najde vhodné očíslování styčníků, lze soustředit nenulové prvky v okolí diagonály a pak je výhodné tuto matici uložit jako pásovou [8]. Dalším vhodným systémem pro uložení matice tuhosti je typ *skyline* případně matice s kompresovanými řádky, sloupci nebo jejich kombinací [86].

3.1.4 Implementace MKP v jazyce C/C++

V této práci je provedena implementace nejen programu MATLAB, ale i v programovacím jazyce C/C++. Pokud jsou k dispozici hotové kódy v programu MATLAB, byla by škoda jich pro implementaci v jazyce C/C++ nevyužít. Pomocí příkazu `ccode` se dají parametricky vyjádřené části kódu, tedy matice tuhosti konstrukce a výpočet vnitřních sil respektive napětí, převést do jazyka C nebo C++ a ty pak použít ve stejném smyslu jako v programu MATLAB. Příkaz `ccode` zajistí i správné přečíslování indexů z `1` až `n` do `0` až `n-1`.

Výpočet neznámých posunů ovšem parametricky vyjádřit nelze kvůli velkému počtu operací při jednotlivých rozkladech matice. Je tedy třeba využít řešič soustavy lineárních rovnic implementovaný v jazyce C nebo C++. Pro plnou matici tuhosti konstrukce byl využit řešič $\mathrm{LDL}^{\mathrm{T}}$ sestavený doc. Kruisem [70], řešič z volně dostupné knihovny LAPACK++ a pro řídkou matici řešič FemCAD.

LAPACK++ (Linear Algebra PACKage in C++) je rozšíření knihovny LAPACKu do C++ [15]. LAPACK je napsaný v jazyce Fortran90 a dají se s ním řešit běžné úlohy numerické lineární algebry, např. řešení soustav lineárních rovnic, problém vlastních čísel a podobně [2]. LAPACK je uzpůsoben k tomu, aby prováděl výpočty rychle a efektivně. Je to standard, který využívají i mnohé komerční programy, jako je MATLAB nebo Mathematica.

FEMCad [85] je řešič soustavy lineárních rovnic pro řídké matice vyvinutý Ing. Vondráčkem. Po zadání matice se nejprve provede analýza jejího typu, aby se mohlo provést optimální přeuspořádání do vhodného tvaru a mohl se vybrat vhodný řešič. Jelikož je matice tuhosti symetrická, stačí uložit do paměti jen její polovinu. Vhodné je též použít metodu kompresovaných řádků, kde uložíme hodnoty jednotlivých prvků dolní trojúhelníkové matice, jejich sloupcové indexy a pozici prvku ve vektoru hodnot, kde začíná nový řádek matice [86].

3.1.5 Soubory MEX pro program MATLAB

Program napsaný v jazyce C nebo C++ bývá ve spoustě případů rychlejší než v jazyce prostředí MATLAB. MATLAB však umožňuje po drobných úpravách takovýto

kód zkompilovat do MEX souboru (*MEX-files*) [81] a poté s ním zacházet stejně jako s vestavěnou funkcí. Pro kompilaci kódu `pokus.c` v jazyce C se do příkazové řádky MATLABu zapíše `mex pokus.c` a tím se vytvoří soubor MEX `pokus.mexw64`. Takovýto MEX soubor však bude spustitelný pouze na stejné verzi operačního systému, ve které byl vytvořen. V této práci je pracováno na 64bitovém operačním systému Windows 7, proto je v příponě vytvořeného souboru `w` jako Windows a `64` jako 64bitový.

Na začátku každého kódu v jazyce C připravovaného pro kompilaci do MEX souboru musí být místo volání hlavní funkce `main` volání `mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray*prhs[])`. Poté musí být na jednom místě definice proměnných používaných v hlavní funkci a též, pokud se používají pole ve funkcích volaných v hlavní funkci, ukazatelé na tato pole, ať už se jedná o pole dynamická nebo statická. Vstupům, se kterými MEX soubor pracuje, musí být přiřazeno pole s reálnými prvky typu `double mxGetPr(prhs[])`, kde `prhs` je pole pointerů na vstupní data. Výstupům musí být přiřazeno pole pointerů `plhs[]`. Výňatek kódu, který je celý zveřejněný v příloze C, ilustruje výše popsaný postup.

```
#include <stdio.h>
#include <math.h>
#include "mex.h"

static neq =8;  // počet nenulových posunů

/* procedura na řešení soustavy lineárních rovnic LDL^T rozkladem */
static void reseni_rovnic(double *a,double *y,double *x,long n,long m,long as) {
  ...
}

/* funkce řešící posuny, napětí a váhu konstrukce pomocí MKP
    A      ... jednorozměrné pole s plochami příčných řezů
    OUT[0] ... maximální posun
    OUT[1] ... maximální napětí
    OUT[2] ... váha konstrukce  */
static void compute ( double A[], double OUT[] ) {
  ...
  reseni_rovnic()
  ...
}

void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray*prhs[] ) {
  /* alokace vstupů, výstupů a pomocných polí */
  double *K, *f, *u, *S, *A, *OUT;

  /* vstupy */
  A = mxGetPr(prhs[0]);    // vytvoření pointeru na data ve vstupu A

  /* výstup */
  plhs[0] = mxCreateDoubleMatrix(3,1,mxREAL);
  OUT = mxGetPr(plhs[0]);
  compute(A,OUT);
  return;
}
```

3.2 Možnosti řešení soustavy lineárních rovnic

Vyřešení soustavy lineárních rovnic je jedním z nejvýznamnějších problémů na poli technických výpočtů [79]. K získání neznámých posunů metodou konečných prvků je třeba takovouto soustavu, tedy $K \cdot r = f$, vyřešit. V tabulce 3.1 je ukázáno, že tento výpočet zabere velkou část výpočetního času. Proto je třeba prozkoumat možné způsoby vyřešení takovéto soustavy.

Ještě je třeba podotknout, že se vektor uzlových zatížení `f` v případě více zatěžovacích stavů změní na matici `F` o rozměru $(n \times l)$, kde n je počet řádků matice tuhosti konstrukce, tedy počet nenulových posunů, a l je počet zatěžovacích stavů. Tím pádem se změní i vektor posunů `r` na matici `R` též o rozměru $(n \times l)$.

3.2.1 Přímé řešiče

Procedura zpětného lomítka implementovaná v programu MATLAB

Je-li definována soustava lineárních rovnic $AX = B$ a je-li třeba získat matici X, pak je v programu MATLAB doporučováno použít proceduru zpětného lomítka (*backslash operator*). Dle typu matice A, tedy zda-li se jedná o matici pásovou, symetrickou, plnou, řídkou apod., se vybere nejvhodnější řešič. Například pro trojúhelníkovou matici se používá substituce, pro čtvercovou plnou symetrickou matici se vybere Choleského dekompozice, pro pásovou symetrickou řídkou matici se vybere speciální pásový řešič v závislosti na šířce pásu atp. [79].

Nevyplatí se používat inverzní matici A^{-1} vedoucí na řešení $X = A^{-1}B$, neboť je tento výpočet náročný na počet operací a vede k zaokrouhlovacím chybám.

LU faktorizace

Tato metoda se dá použít pro každou čtvercovou matici A. Není potřeba ani symetrie ani pásový charakter matice. Základní myšlenkou této metody je rozklad matice A na dolní L a horní U trojúhelníkovou matici tak, že $LU = A$. Vznikne tedy soustava $LUx = b$. Poté se provede substituce $y = Ux$ a následně se vyřeší rovnice $Ly = b$ k získání y a $Ux = y$, čímž se obdrží požadované řešení x [9].

V prostředí MATLAB je tato metoda implementována pod funkcí `lu`.

Choleského dekompozice

Tato metoda vyžaduje, aby matice A byla symetrická pozitivně definitní. Symetrická matice A je taková, pro kterou platí $A^T = A$ a pro její prvky platí, že $a_{ij} = a_{ji}$. Pozitivně definitní matice je taková matice A, pokud pro každý nenulový vektor x platí, že $x^T Ax > 0$ [68]. Choleského dekompozicí se matice A rozloží na horní R a dolní R^T trojúhelníkovou matici. R^T je transponovaná horní trojúhelníková matice. Po rozkladu

je řešena soustava rovnic $R^T Rx = b$, kde se nejprve spočítá soustava $R^T y = b$ k získání pomocného y a $Rx = y$ k získání vektoru resp. matice x [73].

V prostředí MATLAB je tato metoda implementována pod funkcí `chol`.

LDL$^{\mathrm{T}}$ rozklad

Další metodou k výpočtu vektoru x je LDL$^{\mathrm{T}}$ rozklad matice A, kde L je dolní trojúhelníková matice a D je diagonální matice. Matice A musí být opět symetrická pozitivně definitní. K vyřešení soustavy $LDL^{\mathrm{T}}x = b$ je použit pomocný vektor $\bar{b}$, pro který platí $L\bar{b} = b$, kde po přenásobení vzniklé soustavy $LDL^{\mathrm{T}}x = L\bar{b}$ maticí L^{-1} zleva a poté přenásobením D^{-1} zleva vznikne soustava $L^{\mathrm{T}}x = D^{-1}\bar{b}$. Jelikož je matice D diagonální, jsou prvky matice D^{-1} pouze převrácenými hodnotami matice D. L^T je horní trojúhelníková matice, tudíž lze poslední rovnici ze soustavy snadno spočítat, jelikož se jedná o soustavu jedné rovnice o jedné neznámé. Zbytek pak lze dořešit například zpětnou substitucí [36].

V prostředí MATLAB je tato metoda implementována pod funkcí `ldl`.

3.2.2 Metody iterační

Metody iterační se od předchozích zmíněných metod liší v tom, že nezískávají přesné analytické řešení jako metody přímé, ale k přesnému řešení se přibližují posloupností řešení $x^1, x^2, \ldots, x^n$, kde n je poslední iterační krok. Při každém kroku metody je řešení lepší, až se dospěje k řešení postačujícímu. To může být omezeno počtem iteračních kroků případně možnou chybou řešení.

Metoda sdružených gradientů

Metoda sdružených gradientů je příkladem iteračního řešení. Je-li matice A čtvercová symetrická pozitivně definitní, lze použít metodu sdružených gradientů. Odvození této metody je možno nalézt v mnoha zdrojích např. v [68]. Dále bude uveden algoritmus výpočtu dle [73].

Na počátku je voleno $d^0 = r^0 = b - A \cdot x^0$ a vektor x^0. Tento vektor může být buď nulový nebo získaný odhadem. Pokud je však x^0 zvolen špatně, může se počet iterací vedoucí na správný výsledek zvýšit.

Pro $k = 0, 1, 2, \cdots, n$ pak:

$$\alpha_k = \frac{(r^k)^T \cdot d^k}{(d^k)^T \cdot A \cdot d^k}, \tag{3.1}$$

$$x^{k+1} = x^k + \alpha_k \cdot d^k, \tag{3.2}$$

$$r^{k+1} = b - A \cdot x^{k+1}, \tag{3.3}$$

$$\beta_k = \frac{(r^{k+1})^T \cdot A \cdot d^k}{(d^k)^T \cdot A \cdot d^k}. \tag{3.4}$$

$$d^{k+1} = r^{k+1} - \beta_k \cdot d^k, \tag{3.5}$$

Výsledné řešení x^{k+1} je získáno z rovnice 3.2. Hodnota α_k z rovnice 3.1 vyjadřuje optimální krok ve směru d^k. Směry d^k jsou voleny tak, aby byly A-ortogonální, tzn. aby platilo, že $d_i^T \cdot A \cdot d_j = 0$ pro $i \neq j$, což zaručuje rovnice 3.5. Reziduum r^{k+1} je určeno vztahem 3.3 a vyjadřuje, jak daleko je vektor $A \cdot x^{k+1}$ od správné hodnoty vektoru b [68]. Volba β_k zaručuje A-ortogonalitu d^{k+1} vůči d^k.

V prostředí MATLAB je tato procedura k dispozici pod funkcí `pcg`.

3.3 Porovnání jednotlivých výpočetních metod pro testovací konstrukce

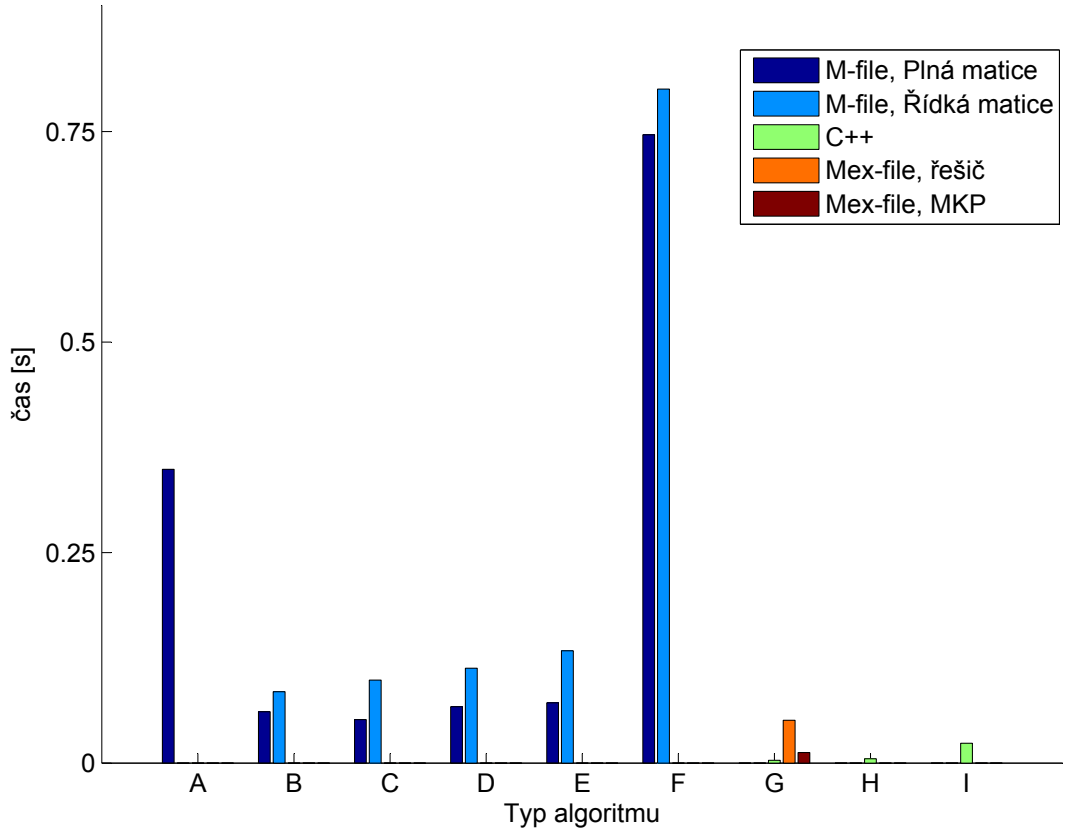
Metody popsané v podkapitolách 3.1 a 3.2 byly zkombinovány a použity pro výpočet hodnot napětí, posunů a hmotností testovacích konstrukcí. Testovací kód, který metody spouští, obsahuje generátor náhodných čísel přiřazující plochy příčných řezů z dané sady na pruty. Jednotlivé skripty byly spuštěny tisíckrát a byla zaznamenána doba tohoto výpočtu. Časová efektivnost jednotlivých metod pro jednotlivé konstrukce je porovnána v grafech 3.2 až 3.5.

Implementace provedená v prostředí MATLAB je zobrazená na sloupcích A až F. Sloupec A vždy znázorňuje metodu, kde se matice tuhosti sestavuje během skriptu. Sloupce B až F naopak znázorňují metody, kde se používají již parametricky vyjádřené matice tuhosti a parametricky vyjádřené vnitřní síly respektive napětí. Zároveň je ve sloupcích B až F vyjádřena matice tuhosti jako plná (tmavěmodrý sloupec) a jako řídká (světlemodrý sloupec). Kódy sestavené v jazyce C/C++ jsou v grafech znázorněny zelenou barvou ve sloupcích G až I. MEX soubor pro řešič soustavy lineárních rovnic je znázorněn oranžovým sloupcem a pro celý kód metody konečných prvků včetně řešiče soustavy lineárních rovnic hnědým sloupcem. Legenda ke grafům je znázorněna v tabulkách 3.3 až 3.6, kde lze zároveň nalézt číselné hodnoty z grafů 3.2 až 3.5.

Řešení soustavy lineárních rovnic bylo v prostředí MATLAB uskutečněno pěti různými způsoby popsanými v podkapitole 3.2. Metoda zpětného lomítka by teoreticky měla být nejrychlejší, neboť je optimalizovaná společností MathWorks. Tato metoda ale nejspíše ztrácí výpočetní čas na posuzování typu matice a výběru správného matematického aparátu. Metoda LU faktorizace se osvědčila pro menší konstrukce, jako je 10-prutová a 25-prutová konstrukce, pro které bylo zároveň výhodné použít matici tuhosti jako plnou. Choleského dekompozice se uplatnila na větších konstrukcích pro

	Prostředí/jazyk Uložení matice Část kódu pro MEX-file	M-file, plná	M-file, řídká	C++	MEX-file, řešič	MEX-file, MKP
A	Sestavení matice za běhu	0,3489				
B	`K * r = f`: Zpětné lomítko	0,0610	0,0848			
C	`K * r = f`: LU faktorizace	0,0516	0,0986			
D	`K * r = f`: Chol. dekompozice	0,0669	0,1127			
E	`K * r = f`: LDL^T rozklad	0,0716	0,1334			
F	`K * r = f`: Metoda s. gradientů	0,7461	0,8004			
G	`K * r = f`: LDL^T rozklad			**0,0033**	0,0509	**0,0125**
H	`K * r = f`: LU faktorizace			0,0051		
I	FemCAD			0,0236		

Tabulka 3.3: Časy výpočtů neznámých na 10-prutové konstrukci pro 1000 spuštění [s]

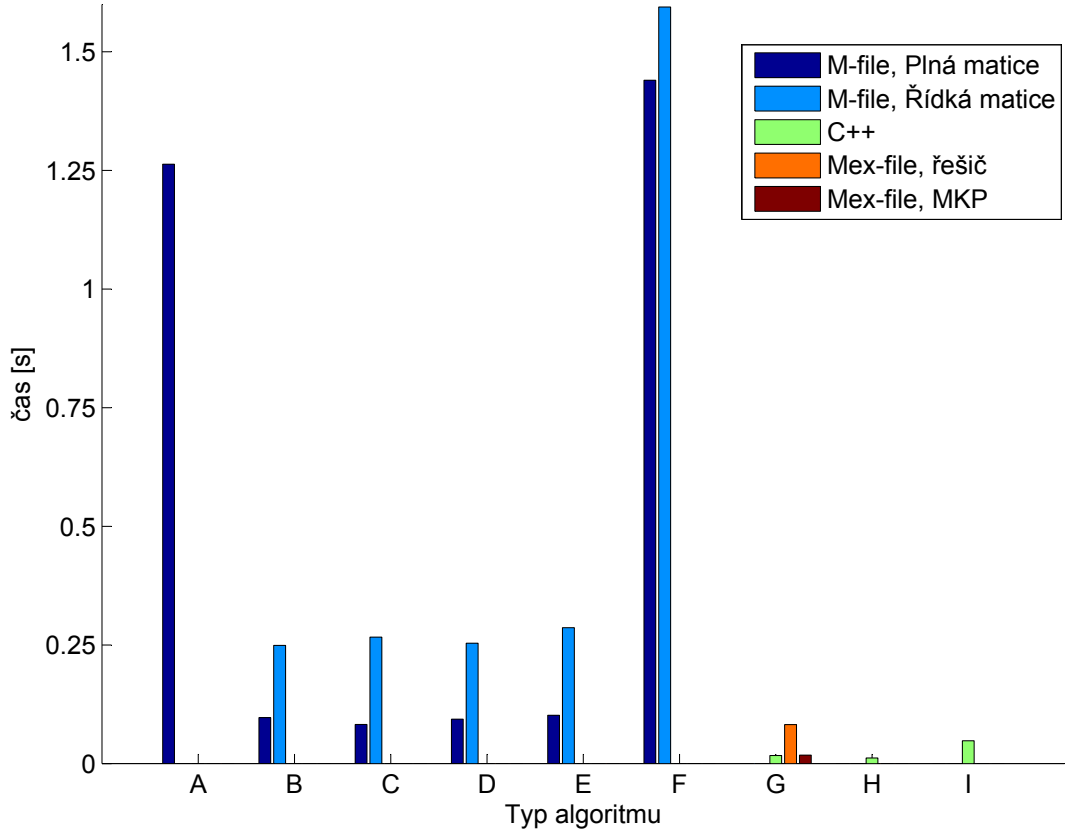


Obrázek 3.2: Graf porovnávající jednotlivé časy výpočtů neznámých na 10-prutové konstrukci, legenda viz tabulka 3.3

plnou matici tuhosti na 52-prutové konstrukci i pro řídkou matici tuhosti pro 72-prutovou konstrukci. LDL^T rozklad, často používaný právě pro tyto typy úloh, byl shledán nejpomalejším. Nemusí se nutně jednat o problém této metody, ale o horší implementaci metody v prostředí MATLAB než u metod ostatních. Metoda konjugovaných gradientů též neobstála, úlohy jsou ještě dostatečně malé a tedy vhodné pro řešiče přímé a nikoliv iterační.

	Prostředí/jazyk Uložení matice Část kódu pro MEX-file	M-file, plná	M-file, řídká	C++	MEX-file, řešič	MEX-file, MKP
A	Sestavení matice za běhu	1,2632				
B	`K * r = f`: Zpětné lomítko	0,0970	0,2492			
C	`K * r = f`: LU faktorizace	0,0824	0,2664			
D	`K * r = f`: Chol. dekompozice	0,0936	0,2536			
E	`K * r = f`: LDL^T rozklad	0,1019	0,2865			
F	`K * r = f`: Metoda s. gradientů	1,4402	1,5945			
G	`K * r = f`: LDL^T rozklad			0,0171	0,0823	**0,0178**
H	`K * r = f`: LU faktorizace			**0,0118**		
I	FemCAD			0,0480		

Tabulka 3.4: Časy výpočtů neznámých na 25-prutové konstrukci pro 1000 spuštění [s]

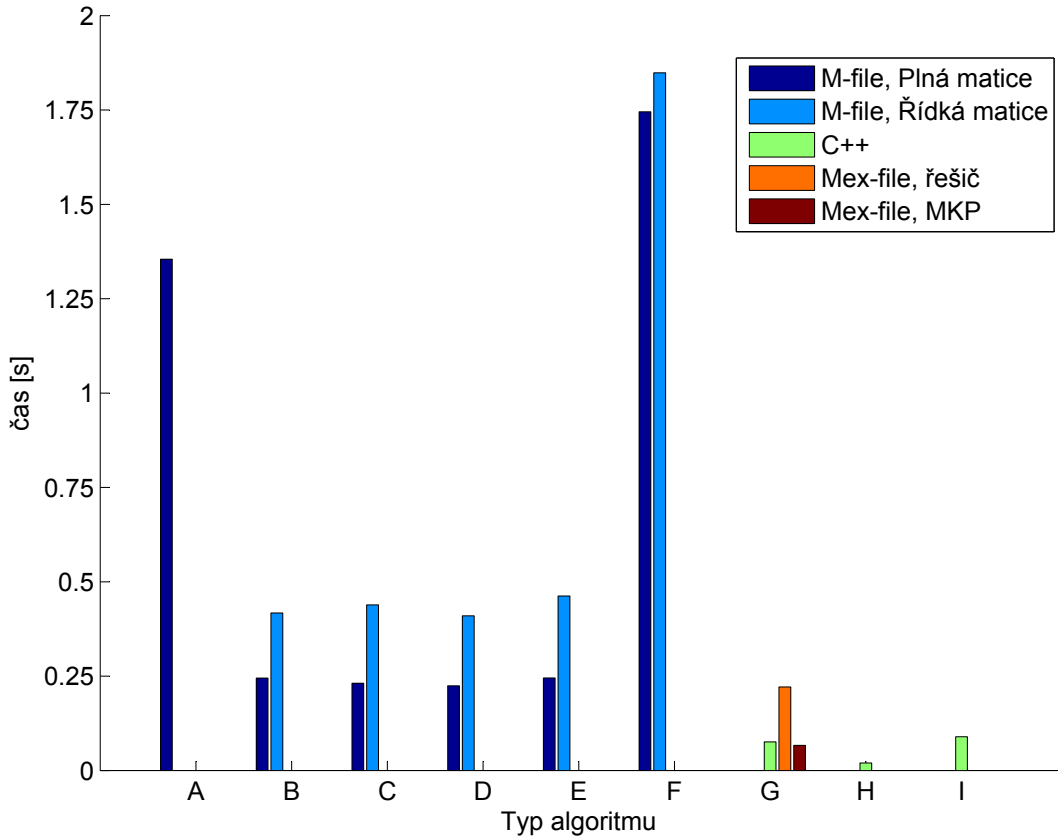


Obrázek 3.3: Graf porovnávající jednotlivé časy výpočtů neznámých na 25-prutové konstrukci, legenda viz tabulka 3.4

Pro úlohu implementovanou v jazyce C/C++ byly použity tři řešiče soustavy lineárních rovnic. LDL^T rozklad se uplatnil u malé 10-prutové konstrukce, u ostatních konstrukcí byl nejrychlejší výpočet pomocí LU faktorizace z balíčku LAPACK++. FemCAD jakožto řešič pro řídké matice neobstál pravděpodobně ze stejných důvodů jako jako metoda konjugovaných gradientů implementovaná v prostředí MATLAB. Úlohy jsou malé a tedy vhodné pro uložení matice tuhosti jakožto plné.

Prostředí/jazyk		M-file,	M-file,	C++	MEX-file,	MEX-file,
Uložení matice		plná	řídká			
Část kódu pro MEX-file					řešič	MKP
A	Sestavení matice za běhu	1,3548				
B	`K * r = f`: Zpětné lomítko	0,245	0,4174			
C	`K * r = f`: LU faktorizace	0,2312	0,4387			
D	`K * r = f`: Chol. dekompozice	0,2245	0,4098			
E	`K * r = f`: LDL^T rozklad	0,2453	0,4624			
F	`K * r = f`: Metoda s. gradientů	1,7452	1,8485			
G	`K * r = f`: LDL^T rozklad			0,0757	0,2214	**0,0666**
H	`K * r = f`: LU faktorizace			**0,0201**		
I	FemCAD			0,0895		

Tabulka 3.5: Časy výpočtů neznámých na 52-prutové konstrukci pro 1000 spuštění [s]

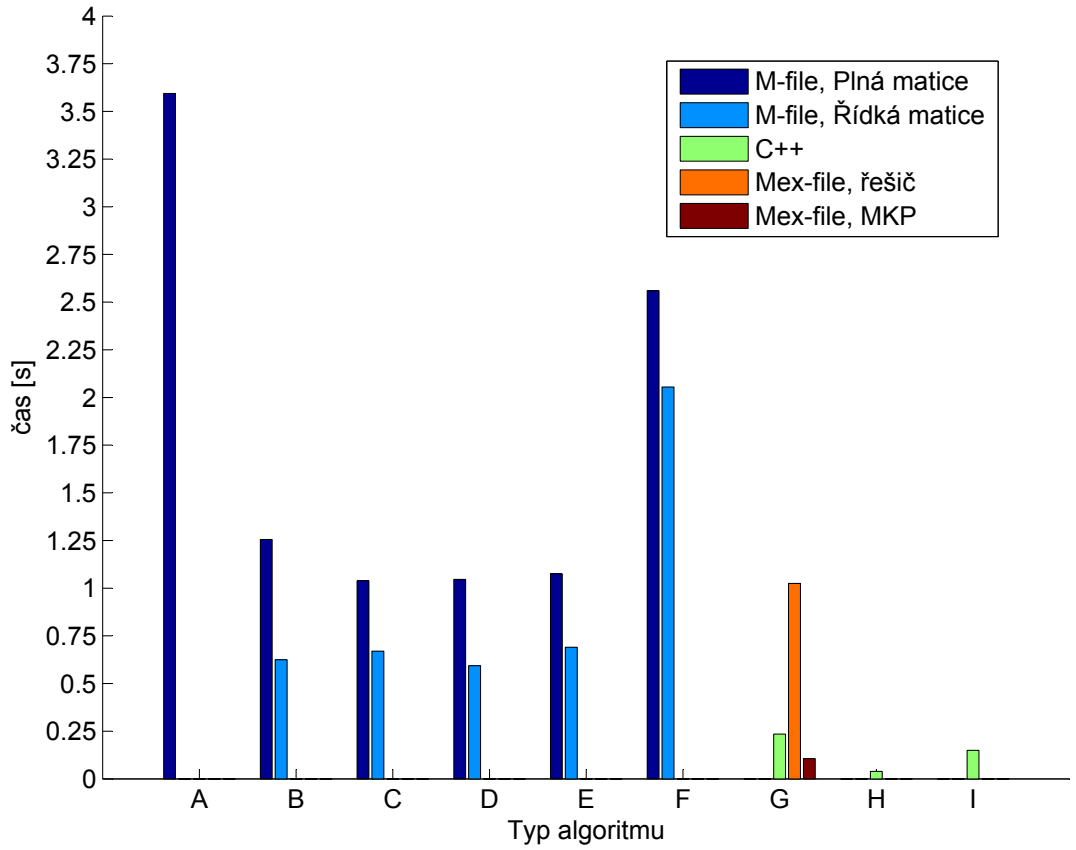


Obrázek 3.4: Graf porovnávající jednotlivé časy výpočtů neznámých na 52-prutové konstrukci, legenda viz tabulka 3.5

MEX soubor použitý pro řešení soustavy lineárních rovnic LDL^T rozkladem dosáhl oproti řešičům implementovaným přímo v prostředí MATLAB malého zlepšení u všech čtyř testovacích konstrukcí, které využívají plné uložení matice. Poté se přistoupilo ke zkompilování celého kódu pro řešení výpočtu MKP. Jediné, co zůstává implementováno v prostředí MATLAB, je generátor náhodných čísel, který přiřazuje plochy příčných řezů z dané sady k jednotlivým prutům. U této metody se dosáhlo výrazného zlepšení,

Prostředí/jazyk		M-file,	M-file,	C++	MEX-file	MEX-file,
Uložení matice		plná	řídká			
Část kódu pro MEX-file					řešič	MKP
A	Sestavení matice za běhu	3,5938				
B	`K * r = f`: Zpětné lomítko	1,2550	0,6246			
C	`K * r = f`: LU faktorizace	1,0383	0,6696			
D	`K * r = f`: Chol. dekompozice	1,0459	0,5931			
E	`K * r = f`: LDL^T rozklad	1,0749	0,6904			
F	`K * r = f`: Metoda s. gradientů	2,5600	2,0546			
G	`K * r = f`: LDL^T rozklad			0,2348	1,0252	**0,1057**
H	`K * r = f`: LU faktorizace			**0,0395**		
I	FemCAD			0,1494		

Tabulka 3.6: Časy výpočtů neznámých na 72-prutové konstrukci pro 1000 spuštění [s]



Obrázek 3.5: Graf porovnávající jednotlivé časy výpočtů neznámých na 72-prutové konstrukci, legenda viz tabulka 3.6

které může směle konkurovat implementaci v jazyce C/C++. V prostředí MATLAB je tato metoda implementace nejlepší u všech čtyř testovacích konstrukcí.

Kapitola 4

Rozměrová optimalizace

4.1 Druhy optimalizací stavebních konstrukcí

Existují různé druhy optimalizačních strategií pro stavební konstrukce. Jednou je třeba navrhnout optimální průřezy na konstrukci, jindy zase navrhnout optimální tvar nosníku. Pro jednodušší orientaci je možné použít dělení dle prof. Stevena [72].

Topologická optimalizace (*Topology optimization*) se zabývá hledáním tvaru konstrukce, který není předem znám, a optimální topologií konstrukce. Je ale známo prostředí (jako např. umístění podpor), optimalizační kritéria (např. hmotnost konstrukce) a omezení [71].

U optimalizace tvaru (*Shape optimization*) je dopředu známá topologie konstrukce, problémy ale může dělat část konstrukce nebo její detail. Snahou je najít optimální tvar, aby byla zajištěna nejlepší distribuce napětí v inkriminovaném místě [7]. Tuto optimalizaci lze použít i na konstrukci jako celek. Je dán základní tvar, ze kterého se vychází, a cílem je najít lepší konstrukci s ohledem na zvolená optimalizační kritéria.

U rozměrové optimalizace (*Size optimization*) je předem zadána sada ploch příčných průřezů, tvar konstrukce, materiál a prostředí, tedy zatížení a podepření. Cílem je zkombinovat průřezy tak, aby byly dodrženy určité předem dané omezující podmínky (maximální deformace konstrukce, maximální napětí apod.) za minimalizace hmotnosti, a tudíž i celkové ceny spotřebovaného materiálu [71]. Rozměrová optimalizace může být spojitá a diskrétní. U diskrétní optimalizace je zadána sada profilů, ze které se vybírá optimální kombinace. Tím pádem je vhodná pro válcované profily, které se vybírají z předem zadaného seznamu výrobce. Úloha však může být formulována i jako spojitá optimalizace. Plochy příčných průřezů pak nabývají jakýchkoliv hodnot z oboru reálných čísel mezi zadanou horní a dolní mezí.

Optimalizace skladby (*Layout optimization*) je speciálním typem rozměrové optimalizace. Pokud se blíží průřezová plocha prvku nulové hodnotě, pak tento prvek nemusí být v konstrukci vůbec použit [42].

4.2 Diskrétní rozměrová optimalizace

Rozměrová diskrétní optimalizace bývá s výhodou využita tam, kde se spojité hodnoty proměnných nedají použít. Jednak to jsou tloušťky plechů, válcované profily vybírané z určitého seznamu, počet šroubů ve spoji nebo třeba počet profilů výztuže v betonovém prvku. Diskrétní optimalizace se může zdát oproti spojité jednodušší, neboť se prohledává jen diskrétní prostor, který je menší. Nicméně kvůli nespojitému prostoru není možné použít rychlé a osvědčené metody matematického programování jako jsou například metody gradientní [90]. K získání globálního optima je nejjednodušším, ale zároveň velice výpočetně náročným způsobem výpočtu metoda hrubé síly (*enumeration*). Efektivnějším nástrojem je pak metoda větví a mezí.

4.2.1 Metoda hrubé síly

Metoda hrubé síly funguje na principu vyčíslení všech možných řešení. Je-li v úloze k proměnných (např. prutů u příhradové konstrukce) a je-li definována sada M, ve které je n možných návrhových prvků (např. válcovaných profilů), pak je celkový počet řešení n^k. S rostoucím problémem, tedy velikostí a složitostí konstrukce (přibývající počet prutů), roste počet vyčíslených řešení exponenciálně. I se zvětšující se sadou, ze které se vybírá, roste problém velmi rychle. Metoda hrubé síly je tedy vhodná jen pro velice malé úlohy kvůli velkým nárokům na výpočetní techniku respektive výpočetní čas.

Metodika výpočtu je ukázána na 5-prutové konstrukci.

```
k=5;   % počet prutů
M=[0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.10];   % sada
len_vek = length(M);   % počet prvků v množině ploch příčných řezů

Lind=ones(1,k);          % dolní hranice - pořadové hodnoty
Uind=len_vek*ones(1,k);  % horní hranice - pořadové hodnoty
res=[Lind,10^10];  % vektor, do kterého se ukládá nejlepší řešení a jeho hmotnost,
                   % na pozici hmotnosti je velké číslo z důvodu minimalizace
X = Lind;

while(X(1)<= Uind(1))
   A = X;
  [true,w]=MKP_5bar(M(A));   % výpočet omezujících podmínek

   % uložení nejlepšího dosud nalezeného řešení
   if (true == 1 && w < res(1,k+1)), res = [A w]; end

   X(k)=X(k)+1;
   for i = k:-1:2
      if (X(i) > Uind(i))
         X(i-1)=X(i-1)+1;
         X(i) = Lind(i);
      end
   end
end
res
```

Na začátku se definuje počet prutů `k` a sada profilů `M`. Jelikož je snažší při vyčíslování řešení pracovat s pořadovými čísly profilů v sadě (jedná se o přirozená čísla), je definována dolní hranice `Lind`[1] jako `1 1 1 1 1` a horní hranice `Uind`[2] jako `10 10 10 10 10`, kde `10` je počet profilů v sadě. Mezi těmito dvěma řešeními se budou vyčíslovat ostatní řešení. Dále je definován vektor `res`, do kterého se bude ukládat dosud nejlepší nalezené řešení a jeho hmotnost. Ve `while` cyklu se postupně odzadu zvětšují hodnoty profilů, tedy začíná se s `1 1 1 1 1`, `1 1 1 1 2` a když se dospěje k `1 1 1 1 10` zvedne se předposlední proměnná o jedničku a poslední se nastaví na svou počáteční hodnotu, tedy `1 1 1 2 1`. Když se na předposlední proměnné nastaví maximální možná hodnota `1 1 1 10 10`, zvedne se 3. proměnná o jedničku `1 1 2 1 1` a takto se pokračuje, dokud není vygenerováno řešení `10 10 10 10 10`. Pro každé vygenerované řešení se spočítá metodou konečných prvků maximální absolutní hodnota napětí a maximální absolutní hodnota posunu ve vodorovném a svislém směru a tyto hodnoty se porovnají s předepsanými maximálními hodnotami v úloze. Pokud jsou obě omezující podmínky splněny, nabyde proměnná `true` hodnoty 1, nejsou-li splněny, je `true` rovno 0. Do proměnné `w` se ukládá hmotnost konstrukce. Řešení konstrukce s nejmenší optimální hmotností, tedy globální optimum úlohy, je na konci úlohy uloženo v `res` a vypsáno na obrazovku.

Jelikož je v této konstrukci dáno pět prutů a deset profilů v sadě, výsledkem je $n^k = 10^5$ možných zadání konstrukce. Vyčíslení všech zadání je tedy možné spočítat v reálném čase. Pro druhou nejmenší, 25-prutovou konstrukci se sadou z 30 profilů, by byl počet zadání konstrukce $n^k = 30^8 = 6,56 \cdot 10^{11}$. Pro tuto úlohu již není možné v reálném čase všechny zadání konstrukce vyčíslit. Nicméně se tato metoda dá použít pro analýzu okolí nějakého řešení, například pro nejlepší optima publikovaná v literatuře. Na obrázku č. 4.2 je v hypergrafu znázorněno okolí globálního optima pětiprutové konstrukce. Globální optimum má skladbu profilů $A_i = (0,05; 0,01; 0,06; 0,02; 0,01)$ in^2, v pořadových číslech `5 1 6 2 1`, a okolí je vytvořeno profily, které se od skladby profilů globálního optima liší o jednu pozici na každé proměnné. V pořadových číslech je tedy zadání konstrukce s nejmenší hmotností `4 1 5 1 1` (na obrázku levý dolní roh) a zadání s největší hmotností `6 2 7 3 2` (na obrázku pravý horní roh). Barevnou škálou jsou rozlišeny hmotnosti konstrukcí. Jednotlivá zadání jsou v pořadových číslech vypsána v obrázku 4.1 a) tak, aby korespondovala s obrázkem 4.2. Zároveň jsou v něm rozlišená zadání, která splňují omezující podmínky (modrou barvou) a která je nesplňují (červenou barvou). Jednotlivé hmotnosti zadání jsou pak vypsány na obrázku 4.1 b) a zároveň graficky znázorněny stejně jako na obrázku 4.2. Jelikož pro větší konstrukce již nebude možné vypisovat hmotnosti číselně, slouží tento obrázek zejména pro vytvoření představy kreslení hypergrafů v této práci.

[1] kombinace profilů s nejmenší možnou hmotností konstrukce

[2] kombinace profilů s největší možnou hmotností konstrukce

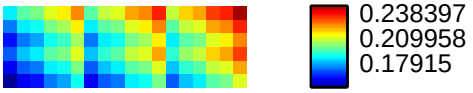
a)

X_2	X_4	X_1=4						X_1=5						X_1=6					
		X_3=5		X_3=6		X_3=7		X_3=5		X_3=6		X_3=7		X_3=5		X_3=6		X_3=7	
		X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2
X_2=2	X_4=3	42531	42532	42631	42632	42731	42732	52531	52532	52631	52632	52731	52732	62531	62532	62631	62632	62731	62732
	X_4=2	42521	42522	42621	42622	42721	42722	52521	52522	52621	52622	52721	52722	62521	62522	62621	62622	62721	62722
	X_4=1	42511	42512	42611	42612	42711	42712	52511	52512	52611	52612	52711	52712	62511	62512	62611	62612	62711	62712
X_2=1	X_4=3	41531	41532	41631	41632	41731	41732	51531	51532	51631	51632	51731	51732	61531	61532	61631	61632	61731	61732
	X_4=2	41521	41522	41621	41622	41721	41722	51521	51522	51621	51622	51721	51722	61521	61522	61621	61622	61721	61722
	X_4=1	41511	41512	41611	41612	41711	41712	51511	51512	51611	51612	51711	51712	61511	61512	61611	61612	61711	61712

b)

X_2	X_4	X_1=5						X_1=6						X_1=7					
		X_3=4		X_3=5		X_3=6		X_3=4		X_3=5		X_3=6		X_3=4		X_3=5		X_3=6	
		X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2	X_5=1	X_5=2
X_2=2	X_4=4	0,179	0,189	0,193	0,203	0,207	0,217	0,189	0,199	0,203	0,213	0,217	0,227	0,199	0,209	0,213	0,223	0,227	0,237
	X_4=3	0,169	0,179	0,183	0,193	0,197	0,207	0,179	0,189	0,193	0,203	0,207	0,217	0,189	0,199	0,203	0,213	0,217	0,227
	X_4=2	0,159	0,169	0,173	0,183	0,187	0,197	0,169	0,179	0,183	0,193	0,197	0,207	0,179	0,189	0,193	0,203	0,207	0,217
X_2=1	X_4=4	0,165	0,175	0,179	0,189	0,193	0,203	0,175	0,185	0,189	0,199	0,203	0,213	0,185	0,195	0,199	0,209	0,213	0,223
	X_4=3	0,155	0,165	0,169	0,179	0,183	0,193	0,165	0,175	0,179	0,189	0,193	0,203	0,175	0,185	0,189	0,199	0,203	0,213
	X_4=2	0,145	0,155	0,159	0,169	0,173	0,183	0,155	0,165	0,169	0,179	0,183	0,193	0,165	0,175	0,179	0,189	0,193	0,203

Obrázek 4.1: Okolí globálního optima 5-prutové konstrukce a) zadání s vyznačením vyhovujících omezujících podmínek b) hmotnosti konstrukce [lb]



Obrázek 4.2: Okolí globálního optima 5-prutové konstrukce (legenda v lb)

4.2.2 Metoda větví a mezí

Dalším možným způsobem obdržení globálního optima je použití metody větví a mezí. Poprvé byla publikována v roce 1960 autorkami A. M. Land a A. G. Doig v [45]. Existují různé modifikace původního algoritmu s využitím jednak pouze diskrétních anebo kombinovaných (tedy spojitých a diskrétních najednou) proměnných. Hlavní myšlenkou je rozdělit problém na podproblémy, tzv. větve, a tím omezit výpočet všech řešení.

Metoda větví a mezí s diskrétními proměnnými

Autor Arora v článku [3] používá pro vysvětlení tohoto algoritmu následující úlohu. Je dána účelová funkce $f(x)$, která má být minimalizována, omezující podmínky ve formě nerovností g_1 až g_3 a sada potenciálních hodnot pro každou proměnnou. Parciální derivace funkce jsou záporné, jde tedy o klesající funkci a jelikož se jedná o minimalizační

úlohu, je výhodné začít s maximálními možným hodnotami proměnných ze sad. Postup úlohy je graficky znázorněn na obrázku 4.3.

$$\text{Minimalizovaná účelová funkce} \quad f = \quad -20x_1 - 10x_2 \tag{4.1}$$

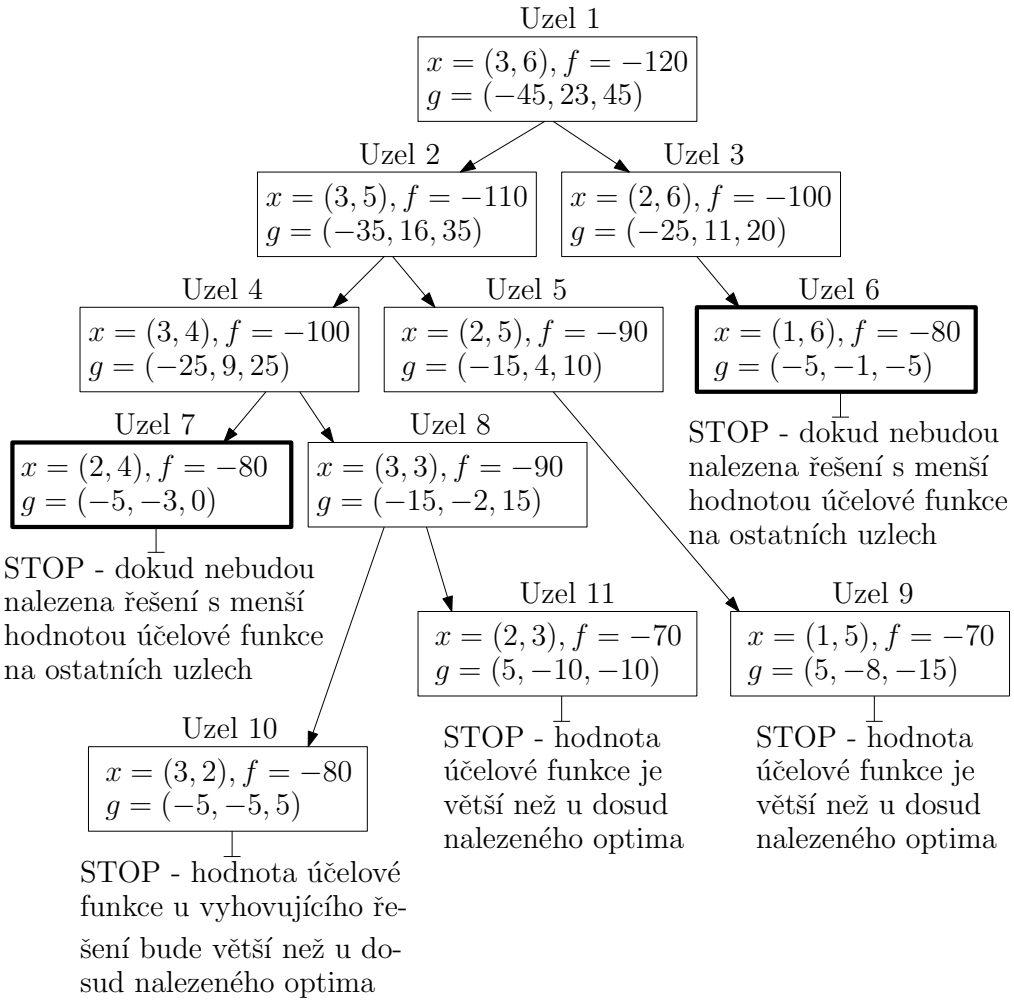
$$\text{s omezujícími podmínkami} \quad g_1 : \quad -20x_1 - 10x_2 + 75 \leq 0, \tag{4.2}$$

$$g_2 : \quad 12x_1 + 7x_2 - 55 \leq 0, \tag{4.3}$$

$$g_3 : \quad 25y_1 + 10y_2 - 90 \leq 0 \tag{4.4}$$

$$\text{a sadami} \quad x_1 \in \quad \{0,1,2,3\}, x_2 \in \{0,1,2,3,4,5,6\}. \tag{4.5}$$

V každém uzlu úlohy, tedy tam, kde se úloha větví, jsou vypočítány hodnoty účelové funkce a omezujících podmínek. Uzel se pak rozvětví tak, že se sníží hodnota jedné proměnné a hodnota té další (respektive dalších u úlohy s více proměnnými) zůstane zachována. Větvení na jednotlivém uzlu končí, pokud jsou splněny omezující podmínky (pak je nalezeno optimum) případně hodnota účelové funkce nabývá vyšších nebo



Obrázek 4.3: Metoda větví a mezí s dikrétními proměnnými

stejných hodnot než je dosud nejlepší nalezené řešení (omezení shora). Uzel 6 a 7 se tedy nevětví, neboť je nalezeno řešení splňující omezující podmínky a při dalším větvení by se hodnota účelové funkce zvyšovala. Uzel 9 a 11 není dále větven, protože je hodnota účelové funkce vyšší než u dosud nalezeného optima a hodnoty omezujících podmínek nejsou splněny. Při dalším větvení k nalezení vyhovujícího řešení by hodnota účelové funkce stále narůstala. U uzlu 10 je hodnota účelové funkce shodná s hodnotou u nalezeného optima, ale omezující podmínky nejsou splněny. Při dalším větvení k nalezení vyhovujícího řešení by ale hodnota účelové funkce narostla. Úloha končí, pokud již nemůže dojít k dalšímu větvení uzlů, v tomto případě je nalezeno globální optimum. Řešením jsou optima v uzlu 6 a 7, která obě splňují omezující podmínky a mají shodnou hodnotu účelové funkce. Z 28 možných kombinací (x_1 může nabýt čtyř různých hodnot, x_2 sedmi hodnot) je nalezeno optimum při 11 kombinacích.

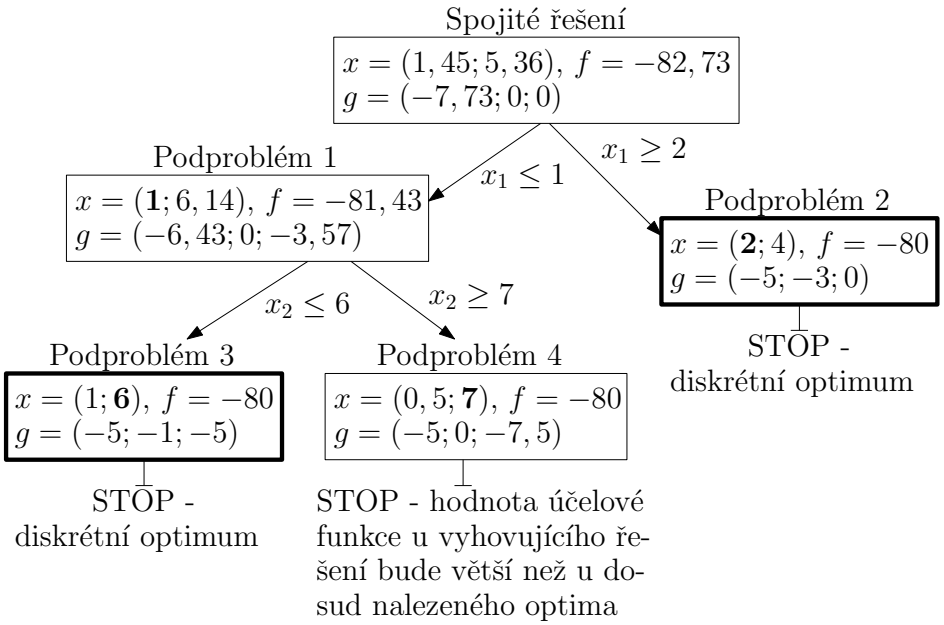
Pokud se jedná o takto jednoduchou úlohu se dvěma proměnnými, není problém se rozhodnout, který uzel větvit, a zajistit, aby se nepočítala zadání již vyřešená. Například uzel 3 by se mohl rozvětvit ještě na druhý uzel, který by ale byl totožný s uzlem číslo 5. Zbytečně by pak narůstala potřeba výpočetního výkonu. Pokud by se větvení nekontrolovalo, úloha by trvala déle. Pokud se naopak větvení kontroluje, je potřeba jednotlivé uzly ukládat a neustále procházet řešení již spočítaná.

Metoda větví a mezí s lokální minimalizací

Pokud je fyzikálně možné, aby proměnné během optimalizace nabývaly spojitých hodnot, pak jako další varianta postupu přichází v úvahu lokální minimalizace s použitím např. metod matematického programování. Pro ukázku se bude řešit úloha zadaná stejně jako v úloze s diskrétními proměnnými [3].

Nejprve se všechny proměnné nastaví jako spojité a použije se některá vhodná metoda matematického programování. Pokud jsou získány hodnoty diskrétní, pak je metoda větví a mezí ukončena. Pokud jsou získány hodnoty spojité, vybere se jedna z nich, na které se úloha rozvětví. Tato proměnná x_i leží mezi dvěma diskrétními proměnnými $d_{ij} < x_i < d_{ij+1}$. Proto se úloha rozdělí na dva podproblémy, větve, kdy se jednomu předepíše přídatná omezující podmínka $x_i \leq d_{ij}$ a druhému $x_i \geq d_{ij+1}$. Tímto rozdělením se eliminuje část spojitého prostoru, ale nevynechá se žádné řešení s diskrétními proměnnými. Podproblémy se opět vyřeší lokální minimalizací. Pokud je získáno řešení s diskrétními proměnnými, podproblém se dále nevětví, omezující podmínky jsou díky lokální minimalizaci vždy splněny. Pokud jsou proměnné s hodnotami spojitými, opět dochází k větvení na další podproblémy. Pokud je hodnota účelové funkce u některého podpoblému větší než u dosud nejlepšího nalezeného řešení, podproblém se již dále nevětví. Úloha končí, pokud již nejsou žádné podproblémy k větvení.

Na obrázku 4.4 se prvotní řešení získá se spojitými proměnnými. Vybere se první proměnná, která leží mezi diskrétními hodnotami 1 a 2 a tento uzel se rozdělí na dva



Obrázek 4.4: Metoda větví a mezí s kombinací spojitých a diskrétních proměnných

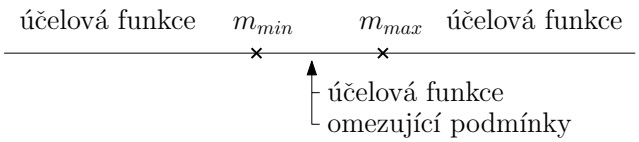
podproblémy s přídatnou omezující podmínkou $x_1 \leq 1$ respektive $x_1 \geq 2$. Lokální minimalizace nalezne dvě možná řešení, kde podproblém 1 obsahuje spojitou proměnnou, tudíž se bude dále větvit, a podproblém 2 obsahuje diskrétní proměnné, tím pádem se ukončí. Lepší řešení z této větve již není možné obdržet. U podproblému 1 se vybere proměnná se spojitou hodnotou a opět dojde k rozdělení na dva podproblémy s omezující podmínkou $x_2 \leq 6$ respektive $x_2 \geq 7$. Lokální optimalizací se opět najdou dvě možná řešení s ohledem na omezující podmínky. Podproblém 3 se ukončí, protože je na obou proměnných dosaženo diskrétních hodnot, podproblém 4 se ukončí, protože hodnota proměnné 7 neleží v zadané sadě a hodnota účelové funkce by při dalším větvení byla větší než u nalezeného diskrétního optima. Nalezená optima jsou stejná jako v předchozím řešení a to v pěti krocích.

U tohoto příkladu je jasné, jak se větvení problému na podproblémy provede. Pokud je však proměnných více, vybírání uzlu k větvení je možné několika způsoby a pokud se provede špatně, metoda selže. Rovněž není u této metody zaručeno získání globálního optima pro všechny typy úloh, protože se větev může ukončit dříve, než je globálního optima dosaženo.

Modifikovaná metoda větví a mezí s diskrétními proměnnými

V ohledu na výše dvě zmiňované metody bylo třeba použít robustnější metodu, která by byla schopná globální optimum dohledat. Zároveň byla tato metoda vyvíjena rovnou pro testovací konstrukce uvedené v kapitole 2, tudíž by měla být dostatečně specializovaná. Nicméně je použitelná i pro jiné typy konstrukcí respektive úloh.

Modifikovaná metoda větví a mezí je postavena na principu metody hrubé síly se zavedením horní m_{max} a dolní m_{min} meze hodnoty účelové funkce. Dolní mez je možné dopočítat spojitou optimalizací anebo odhadnout na základě znalostí publikovaných v literatuře. Hodnota účelové funkce globálního optima se spojitými proměnnými bude vždy menší nebo rovna než u globálního optima s proměnnými diskrétními, neboť je možné najít takové spojité proměnné, pro které bude platit rovnost v omezujících podmínkách. Pokud se bude jednat o odhad, musí být tato hodnota také menší. Horní mez je možné získat například některou heuristickou metodou s diskrétními proměnnými, která dohledá optimum lokální. Hodnota horní meze pak bude vyšší než hodnota globálního optima anebo tomuto optimu rovna. V enumeraci se vyčíslovaly omezující podmínky u všech zadání, což je výpočetně velmi náročné. V modifikované metodě větví a mezí se tyto omezující podmínky ve formě přípustného napětí a posunu vyčíslují (pomocí metody konečných prvků) pouze mezi těmito dvěma mezemi, jak je znázorněno na obrázku 4.5. Hodnota účelové funkce se pak cíleně počítá pouze pro část řešení, pro které je to zapotřebí.



Obrázek 4.5: Zadání konstrukce seřazená dle velikosti

Hodnoty účelové funkce je zbytečné počítat pro všechna zadání. Stejně jako v případě metody hrubé síly se hodnoty proměnných postupně navyšují. Je-li např. na zadání `10 2 1 1 3` dosaženo vyšší hodnoty účelové funkce než je horní mez, pak je zbytečné počítat hodnotu účelové funkce i pro zadání konstrukce s profily `4` až 10 na posledním prutu. Hmotnost konstrukce bude s těmito profily vždy větší. Stejně tak tomu je i pro dolní mez. Pokud je jasné, že hodnota účelové funkce bude nižší než dolní mez i pro konstrukci, která má největší profil na posledním prutu, např. `1 1 1 1 10`, není nutné počítat hmotnost u konstrukcí se zadáním profilů u prutu posledního s pořadovými čísly `1` až `9`.

Čím jsou horní a dolní mez přesnější, tedy blíže k hodnotě globálního optima, tím je menší prostor, který se cíleně prohledává. V průběhu výpočtu je tedy vhodné horní mez upravovat dle dosud nejlepšího nalezeného řešení, protože je zbytečné počítat omezující podmínky i pro řešení s vyšší hmotností konstrukce. Např. na zadání s pořadovými čísly profilů `5 1 6 2 1` byla hmotnost mezi dolní a horní mezí a zároveň byly splněny omezující podmínky. Původní horní mez se snížila z hodnoty 0,23 lb na 0,179 lb. Zadání s profilem o jeden větší na posledním prutu je při snížení horní meze již nevyhovující, protože leží již právě nad horní mezí (zadání `5 1 6 2 2` s hmotností 0,189 lb). Tím se prostor znatelně zmenšil.

Č.	P_i	m	Č.	P_i	m	Č.	P_i	m	Č.	P_i	m
1	1 1 1 1 1	0,058	48,58	1 1 1 6 10	0,198	202	1 1 2 9 9	0,232		⋮	
	1 1 1 1 2	0,068	60	1 1 1 7 1	0,118	193, -	1 1 2 9 10	0,242	11745	4 10 2 2 1	0,240
	⋮			⋮		203,204	1 1 2 10 1	0,162		⋮	
2	1 1 1 1 10	0,148	64	1 1 1 7 5	0,158		⋮		11746	4 10 3 1 1	0,240
5	1 1 1 2 1	0,068		⋮		210	1 1 2 10 7	0,222		⋮	
6	1 1 1 2 2	0,078	59,69	1 1 1 7 10	0,208	211	1 1 2 10 8	0,232	11747	5 1 1 1 1	0,240
7	1 1 1 2 3	0,088	71	1 1 1 8 1	0,128		1 1 2 10 9	0,242		⋮	
8	1 1 1 2 4	0,098	72	1 1 1 8 2	0,138		1 1 2 10 10	0,252	12143,12144	5 1 6 1 1	0,169
9	1 1 1 2 5	0,108	73	1 1 1 8 3	0,148		⋮			⋮	
10	1 1 1 2 6	0,118	74	1 1 1 8 4	0,158	11708,11709	4 9 1 1 1	0,201	12151	5 1 6 1 8	0,239
11	1 1 1 2 7	0,128		⋮			⋮			⋮	
12	1 1 1 2 8	0,138	70,80	1 1 1 8 10	0,218	11712	4 9 1 1 4	0,231	**12152,12153**	**5 1 6 2 1**	**0,179**
13	1 1 1 2 9	0,148	82	1 1 1 9 1	0,138		4 9 1 1 5	0,241	12154	5 1 6 2 2	0,189
3,4,14	1 1 1 2 10	0,158	83	1 1 1 9 2	0,148		⋮			⋮	
11	1 1 1 3 1	0,078	84	1 1 1 9 3	0,158	11713,11714	4 9 1 2 1	0,211	12155	5 1 6 3 1	0,189
	⋮			⋮		11715	4 9 1 2 2	0,221		⋮	
24	1 1 1 3 9	0,158	81,91	1 1 1 9 10	0,228	11716	4 9 1 2 3	0,231	13254	10 2 1 3 1	0,182
15,25	1 1 1 3 10	0,168	93	1 1 1 10 1	0,148		⋮			⋮	
27	1 1 1 4 1	0,088	94	1 1 1 10 2	0,158	11717,11718	4 9 1 3 1	0,221	13255,13256	10 2 2 1 1	0,177
	⋮			⋮		11719	4 9 1 3 2	0,231	13257	10 2 2 1 2	0,187
34	1 1 1 4 8	0,158	101	1 1 1 10 9	0,228		⋮			⋮	
35	1 1 1 4 9	0,168	92,102	1 1 1 10 10	0,238	11720	4 9 1 4 1	0,231	13258	10 2 2 2 1	0,187
26,36	1 1 1 4 10	0,178	103,105	1 1 2 1 1	0,238		⋮			⋮	
38	1 1 1 5 1	0,098		⋮			4 9 1 10 10	0,331	13259	10 2 3 1 1	0,187
	⋮		113	1 1 2 1 9	0,152	11721,11722	4 9 2 1 1	0,216		⋮	
44	1 1 1 5 7	0,158	104,114	1 1 2 1 10	0,162	11723	4 9 2 1 2	0,226	13260,13261	10 3 1 1 1	0,177
45	1 1 1 5 8	0,168	116	1 1 2 2 1	0,082	11724	4 9 2 1 3	0,236	13262	10 3 1 1 2	0,187
46	1 1 1 5 9	0,178		⋮			⋮			⋮	
37,47	1 1 1 5 10	0,188	124	1 1 2 2 9	0,162	11725,11726	4 9 2 2 1	0,226	13263	10 3 1 2 1	0,187
49	1 1 1 6 1	0,108	115,125	1 1 2 2 10	0,172	11727	4 9 2 2 2	0,236		⋮	
	⋮			⋮			⋮		13264	10 3 2 1 1	0,187
54	1 1 1 6 6	0,158	194	1 1 2 9 1	0,152	11728	4 9 2 3 1	0,236		⋮	
55	1 1 1 6 7	0,168	195	1 1 2 9 2	0,162		⋮		13265	10 4 1 1 1	0,187
56	1 1 1 6 8	0,178		⋮		11742,11743	4 10 2 1 1	0,230		⋮	
57	1 1 1 6 9	0,188	201	1 1 2 9 8	0,222	11744	4 10 2 1 2	0,240		10 10 10 10 10	0,583

počítá se ÚF, je pod DM
nepočítá se ÚF, je pod DM
počítají se ÚF a OP, mezi DM a HM, OP nejsou splněny
počítají se ÚF a OP, mezi DM a HM, OP jsou splněny
počítá se ÚF, je nad HM
nepočítá se ÚF, je nad HM

ÚF - účelová funkce, OP - omezující podmínky, DM - dolní mez, HM - horní mez

Obrázek 4.6: Vypsání zajímavých zadání konstrukce včetně vypočtené hmotnosti w v [lb], Č. je číslo kroku, P_i je zadání konstrukce, na začátku nastavena dolní mez $m_{min} = 0,172$ lb a horní mez $m_{max} = 0,23$ lb

Na obrázku 4.6 jsou pak vypsána zajímavá zadání a jejich okolí na 5-prutové konstrukci. Šedou barvou jsou znázorněna zadání ležící pod dolní mezí, modrou barvou zadání nad horní mezí. Omezující podmínky se počítají mezi těmito dvěma mezemi, červeně znázorněná zadání jsou ta, u kterých omezující podmínky nejsou splněny, zelenou barvou pak tak s vyhovujícími omezujícími podmínkami. Pokud jsou v tabulce tři tečky, znamená to, že jsou v tabulce některá zadání vynechána. Jsou-li tato vynechaná zadání označena barvou z legendy, pak se se všemi provádí operace znázorněná touto

barvou. Jsou-li ponechána bílá, pak jsou s nimi prováděny kombinace operací z legendy.

Jako dolní odhad meze účelové funkce pro metodu větví a mezí je použita hodnota hmotnosti ze spojité optimalizace. Jako horní odhad meze byla zvolena hmotnost 0,23 lb, což je zhruba o 25 % více než je hmotnost konstrukce s optimální skladbou prutů. Mezi těmito dvěma mezemi se prostor systematicky prohledává, až se dospěje ke globálnímu optimu.

Algoritmus řešení lze popsat v těchto krocích [62].

1. Nejprve je potřeba rozhodnout, které hodnoty ze sady profilů se použijí jako výchozí. Jelikož se jedná o minimalizaci účelové funkce, tedy hmotnosti konstrukce, je vhodné začít s nejmenšími plochami a ty pak postupně zvětšovat. Jak už bylo řečeno, pro úlohu programování je jednodušší používat celočíselné proměnné, které budou pořadovými hodnotami ze sady profilů - množiny M. Číslování jednotlivých prutů viz obrázek 2.5.

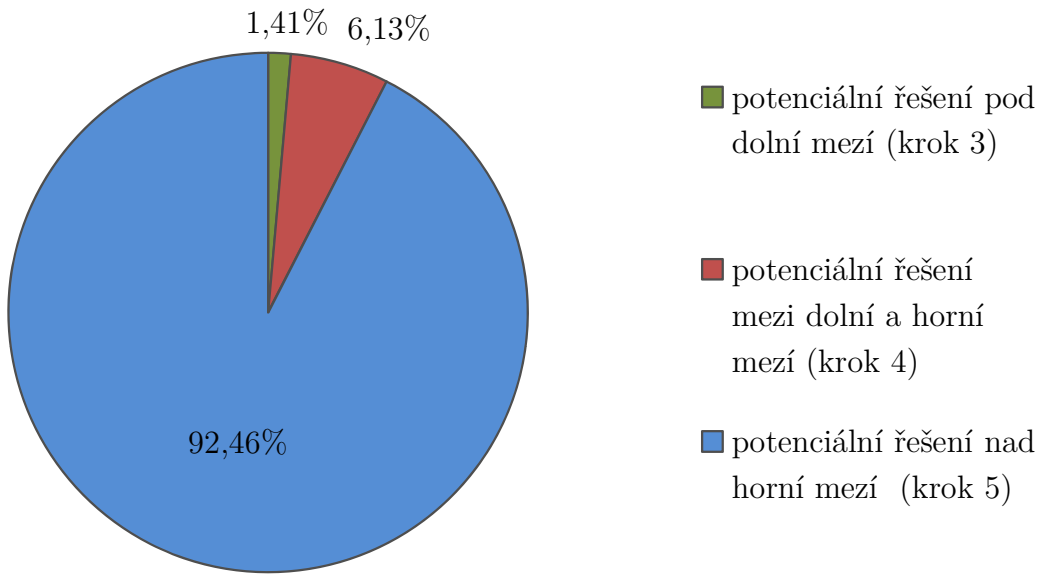
2. Dále se zjistí hodnota účelové funkce - hmotnost m a porovná se s horní mezí m_{max} a dolní mezí m_{min}. Je-li hmotnost m menší než m_{min}, další postup úlohy viz bod 3, je-li hmotnost m mezi hodnotami m_{min} a m_{max}, další postup úlohy viz bod 4 a je-li hmotnost m větší než m_{max}, další postup úlohy viz bod 5.

3. Hodnota m je menší než m_{min}. Je třeba nalézt takovou kombinaci proměnných, kdy bude tato hodnota větší. Pro rychlejší postup úlohy se poslední hodnota proměnné zvedne na maximum, tedy např. `1 1 1 1 10` a vypočítá se hodnota účelové funkce m.

   (a) Je-li hmotnost m této kombinace ploch na konstrukci menší než m_{min}, zvedne se předposlední proměnná o jedničku, tedy na `1 1 1 2 10`, spočítá se hmotnost konstrukce m a opět se porovná s m_{min}. Pokud je tato hmotnost opět menší, zvedne se předposlední proměnná opět o jedničku a takto se pokračuje, dokud hmotnost m není větší než m_{min}. Pokud hodnota předposlední proměnné nabude svého maxima, sníží se opět na své minimum a 3. proměnná od konce se zvedne o jedničku atd. Ve chvíli, kdy jsou proměnné nastaveny tak, že $m > m_{min}$, se pokračuje bodem 2.

   (b) Je-li hmotnost m této kombinace větší než m_{min}, pak se hodnota poslední proměnné sníží na své minimum (tedy např. z `1 1 1 2 10` na `1 1 1 2 1`) a zvyšuje se postupně po jedné (tedy nejprve `1 1 1 2 2`, dále `1 1 1 2 3`, ..., `1 1 1 2 9`, atd.), dokud není $m > m_{min}$. Poté se pokračuje bodem 2.

4. Hodnota m je větší než m_{min} a menší než m_{max}. V tomto podprostoru úlohy bude někde ležet globální optimum úlohy. Je třeba spočítat hodnotu omezujících podmínek, tedy maximální posun ve vodorovném a svislém směru $\max|w_j|$ a hodnotu maximálního napětí $\max|\sigma_i|$. Tyto omezující podmínky se vyhodnotí.



Obrázek 4.7: Graf s rozdělením potenciálních řešení pro 5-prutovou konstrukci

(a) Jsou-li podmínky splněny, tedy $\max|\sigma_i| \leq 60$ ksi a $\max|w_j| \leq 0,06$ in, pak se přepíše horní mez účelové funkce na $m_{max} = m$. Tím se znatelně zmenší prohledávaný prostor úlohy, neboť je horní mez postupně stlačována dolů k hodnotě globálního optima. Dále se poslední proměnná zvedne o jedničku (změní-li se tato proměnná nad maximální hodnotu ze sady např. `1 1 5 11 1`, zvedne se hodnota proměnné před ní a proměnná, která nabyla vyšší hodnoty než je v sadě, se nastaví opět na minimum - `1 1 6 1 1`). Dále viz bod 2.

(b) Nejsou-li tyto podmínky splněny, pak se hodnota poslední proměnné zvedne o jedničku a pokračuje se bodem 2.

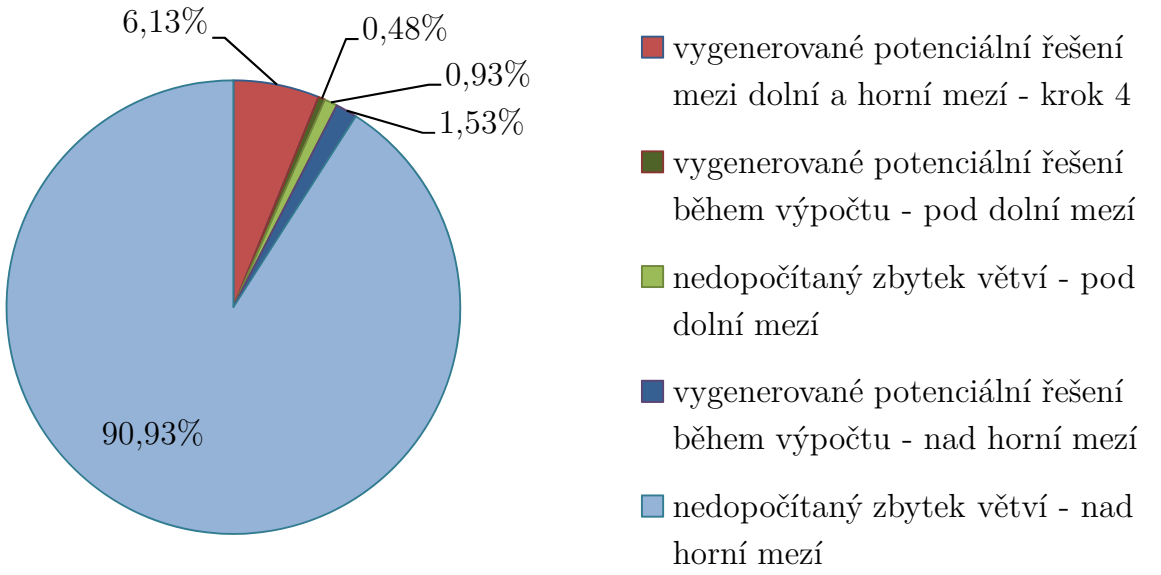
5. Hodnota m je větší než m_{max}. Poslední proměnná se sníží na minimum, předposlední se zvedne o jedničku a spočítá se hodnota účelové funkce. V případě změny proměnné nad maximální hodnotu ze sady se algoritmus chová, jak už bylo popsáno např. v bodě 4a.

   (a) Je-li hodnota m menší než m_{max}, pak se pokračuje bodem 2.

   (b) Je-li hodnota m větší než m_{max}, pak se zvedne 3. proměnná od konce o jedničku a předposlední se nastaví na své minimum. V tomto duchu se pokračuje, dokud hodnota účelové funkce příslušné kombinace m není menší než m_{max}. Pokud takováto kombinace neexistuje, úloha končí.

6. Při nastavení všech proměnných na své maximum úloha končí.

Na obrázku 4.7 je znázorněno rozložení jednotlivých potenciálních řešení problému podle toho, do které části obrázku 4.5 spadají. Na obrázku 4.8 je pak vyjádřeno, kolik



Obrázek 4.8: Graf s rozdělením potenciálních řešení pro 5-prutovou konstrukci

jednotlivých řešení se prochází, tj. počítá se u nich hodnota účelové funkce, a u kolika se účelová funkce nepočítá. Konečné výsledky lze najít v kapitole 6, kde je srovnání výsledků jednak spojité optimalizace a jednak diskrétní optimalizace pomocí hrubé síly a metody větví a mezí.

4.3 Spojitá rozměrová optimalizace

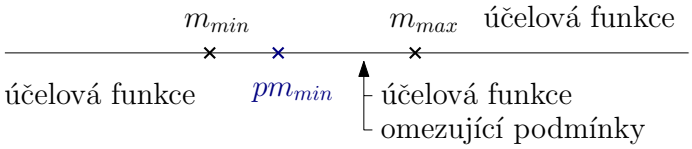
Úloha spojité optimalizace se zdá být komplexnější a na řešení náročnější než úloha diskrétní, neboť se řešení vyhledává ve větším prostoru. Tento prostor je složený z reálných čísel a je ohraničen mezemi jednotlivých proměnných. Naproti tomu diskrétní prostor je složen z konečného počtu kombinací daných hodnot v zadání úlohy. Ale právě díky tomu, že je tento prostor spojitý, je možné použít úlohy matematického programování, které jsou v dnešní době již velmi dobře prozkoumané [29]. Tyto metody jsou rychlé a většinou velmi úspěšné, nicméně nezaručují nalezení globálního optima. To však není u spojitých proměnných možné zaručit u žádné metody právě kvůli nekonečně velkému prostoru a tedy nekonečně velkému počtu řešení.

Jelikož se ale u modifikované metody větví a mezí znalost spojitého globálního optima vyžaduje pro nastavení dolní meze, je třeba zajistit alespoň hodnotu účelové funkce nižší než je toto globální optimum i za možnosti nesplnění omezujících podmínek. Tím sice naroste potřeba výpočetního výkonu, ale bude zajištěna kvalita prezentovaného řešení s diskétními proměnnými.

Prvním způsobem, který zaručuje, že nalezené optimum pro diskrétní proměnné bude globální, je vynechání spojité optimalizace a použití dolní meze jako nejnižší možné hmotnosti konstrukce, viz obrázek 4.9. U úlohy pětiprutové konstrukce tedy

Obrázek 4.9: Zadání konstrukce seřazená dle velikosti s dolní mezí bez spojité optimalizace

bude zadání $A_i = (0,01; 0,01; 0,01; 0,01; 0,01)$ in^2 a jeho hmotnost 0,058 lb. Tímto ale rapidně naroste potřeba výpočetního výkonu respektive výpočetního času.



Obrázek 4.10: Zadání konstrukce seřazená dle velikosti s odhadnutou dolní mezí (pm_{min} je původní předpoklad globálního optima spojité optimalizace)

Druhým možným způsobem, který ovšem nezaručuje stoprocentní jistotu nalezení globálního diskrétního optima, nicméně zdá se být dostatečně dobrým, je odhadnutí globálního optima spojité úlohy případně hodnoty o něco málo nižší. Kvůli výpočtu omezujících podmínek je daná úloha nelineární. V prostředí MATLAB je v *Optimization Toolbox*u implementovaná úloha podmíněného nelineárního programování `fmincon()`. Pokud je tato metoda spuštěna z několika náhodných bodů[3] a výsledky nevykazují velké odchylky ani mezi sebou ani od výsledků prezentovaných v dostupné literatuře, je možné vzít tyto výsledky jako uspokojivé a tedy dostatečně spolehlivé. Pokud však tyto výsledky rozdíly vykazují, a to jak mezi sebou, tak od prezentovaných výsledků v publikované literatuře, nelze úlohu nelineárního programování pro výpočet dolní meze spolehlivě použít. Pak je nasnadě použít dostatečně dobrý odhad dolní meze, a to např. snížením hodnoty nalezené v publikované literatuře např. o 20 %. Toto snížení bude již tak velké, že bude zachována kvalita prezentované modifikované metody větví a mezí v kompromisu s potřebou výpočetního času. Jak se úloha zvětší je možné vidět na obrázku 4.10.

4.3.1 Nelineární programování v prostředí MATLAB

Funkce nelineárního programování `fmincon()` [75] nabízí možnost výběru ze čtyř možných optimalizačních metod. Hlavní myšlenkou prvního algoritmu `trust-region-reflective` je aproximovat účelovou funkci $f(x)$ jednodušší funkcí q, která má podobné chování na okolí bodu x. Toto okolí se nazývá *trust region*, v překladu důvěryhodná oblast. Ve standardní implementaci algoritmu se pro jednodušší funkci q používá kva-

[3] při výpočtu totiž hrozí riziko pádu do lokálního optima

dratická aproximace, která je definovná prvními dvěma členy Taylorova rozvoje. Této metodě je však nutné dodat gradient účelové funkce a pro omezující podmínky nesmí být použity nerovnosti [76]. Další metodou je `active-set`, která používá k řešení Karush-Kuhn-Truckerovy podmínky nutné pro existenci minima podmíněné optimalizace. Zároveň je zde opět využito sekvenčního kvadratického programování, kde se v každém kroku iterace řeší podproblém kvadratickým programováním. Třetí metodou je `interior-point`, který řeší minimalizační úlohu převodem na posloupnost přibližných úloh. Omezující podmínky ve formě nerovností se převedou na rovnosti. K vyřešení přibližného problému je použit buď Newtonův krok anebo metoda sdružených gradientů. Čtvrtá, nejnovější metoda `sqp` využívá sekvenční kvadratické programování. Je velmi podobná druhé metodě. Rozdíly jsou ale zejména v: použití mezí; `sqp` se pohybuje při každé iteraci pouze v oblasti omezené mezemi. `sqp` též využívá odlišné výpočetní metody lineární algebry k řešení podproblému kvadratického programování než `active-set`, kde `sqp` kombinuje objektivní a omezující funkce do jedné, čímž se zlepší pravděpodobnost získání optima, ale zároveň narůstá počet proměnných a výpočet může trvat déle. Pokud nejsou splněny omezující podmínky, využívá se aproximace 2. řádu jejich funkcí, což opět může zpomalit výpočet ovšem za cenu možného zlepšení řešení.

Funkci `fmincon()` je třeba na začátku zadat startovací vektor, účelovou funkci, omezující podmínky a hodnoty horních a dolních mezí neznámých. Pomocí funkce náhodného rovnoměrného rozdělení je možné získat rozdílné startovací body, metodu spustit například stokrát a vybrat nejlepší získané řešení. Účelovou funkcí, která má být minimalizována, je hmotnost konstrukce $f(A_i) = m = \rho \cdot \sum A_i \cdot L_i$, kde i je číslo prutu a nabývá hodnot $i = 1, \ldots, k$, kde k je celkový počet prutů, ρ je objemová hmotnost materiálu, A_i je plocha příčného řezu a L_i je délka prutu. Omezující podmínky jsou nerovnice, a to $\max|\sigma_i| - \sigma_{lim} \leq 0$ a $\max|w_j| - w_{lim} \leq 0$, kde $\max|\sigma_i|$ je maximální absolutní hodnota napětí, σ_{lim} je maximální dovolená hodnota napětí daná v zadání úlohy, $\max|w_j|$ je maximální absolutní hodnota posunu ve vodorovném i svislém směru, w_{lim} je maximální dovolená hodnota posunu v zadání úlohy a j je pořadové číslo volných posunů. Hodnoty $\max|w_j|$ a $\max|\sigma_i|$ je možné spočítat využitím již implementované metody konečných prvků. Výsledkem je pak hmotnost konstrukce m jakožto hodnota minimalizované účelové funkce a vektor skladby jednotlivých ploch na prutech.

Kód programu by mohl vypadat například takto. Zadají se horní `lb` a dolní mez `ub` a vygeneruje se náhodný startovací bod `x0`. Nastaví se optimalizační metoda pomocí `options` a poté se spustí vlastní optimalizace, kde výstupem je vektor zadání `x` a hodnota účelové funkce `fval`. `fmincon` volá `objfun.m` pro výpočet účelové funkce a `confun.m` pro výpočet omezujících podmínek.

```
%% run.m
lb = 0.01*ones(1,5);          % nastavení dolní meze
ub = 0.1*ones(1,5);           % nastavení horní meze
x0 = rand(1,5).*(ub-lb)+lb;  % náhodný startovací bod

options = optimset('Algorithm','active-set','Display','off');
[x,fval]=fmincon(@objfun,x0,[],[],[],[],lb,ub,@confun,options) % vlastní optimalizace
```

Účelová funkce je výpočet hmotnosti konstrukce, kde vstupem je vektor ploch `Area` a výstupem je hmotnost konstrukce `w`.

```
%% objfun.m
function w = objfun(Area)
  L = [10,14.1421356237310,14.1421356237310,10,10]; % délka prutů [in]
  w=0.1*sum(Area*L');                               % hmotnost konstrukce [lb]
end
```

Omezující podmínky je třeba zadat ve formě nerovnosti `c` a rovnosti `ceq`. Pokud například rovnosti chybí, je nezbytné zadat prázdný vektor `ceq = []`. Hodnoty maximálního napětí a maximálního posunu se spočítají metodou konečných prvků, jak je popsáno v příloze B nebo kapitole 3. Výstupem pro `fmincon` je vektor, který obsahuje omezující podmínky ve tvaru $c(x) \leq 0$, tzn. od maximální hodnoty napětí respektive posunu se musí odečíst jejich maximální povolená hodnota definovaná v zadání úlohy.

```
%% confun.m
function [c, ceq] = confun(Area)
  % Nelineární podmínka rovnováhy - nerovnost
    E  = 10^4;                                          % modul pružnosti [ksi]
    L = [10,14.1421356237310,14.1421356237310,10,10]; % délka prutů [in]
    ki = E*Area./L;                                     % tuhost prutů
    Disp = [...];                                       % výpočet posunů
    Mforces = [...];                                    % výpočet vnitřních sil

    maxs=max(max(abs(Mforces./Area')));                 % maximální napětí
    maxw=max(max(abs(Disp)));                           % maximální posun

    c = [maxs-60; maxw-0.06];     % Výstupní vektor omezujících podmínek

  % Nelineární podmínka rovnováhy - rovnost
    ceq = [];
end
```

Kapitola 5

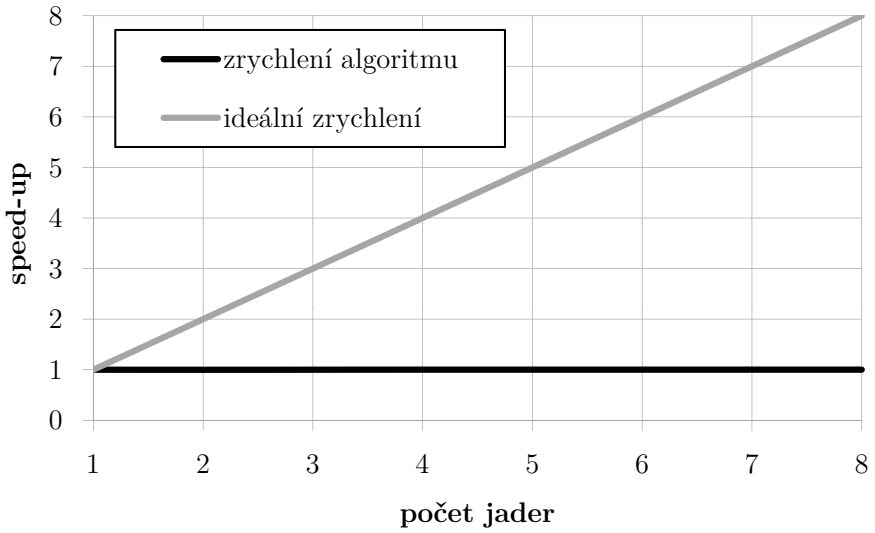
Paralelizace

Výpočty pomocí metody větví a mezí jsou velmi náročné na dobu výpočtu. Jednotlivé kroky jsou však na sobě relativně nezávislé, proto je vhodné využít paralelního způsobu výpočtu na více procesorech respektive jádrech procesoru najednou. Jediná část úlohy, která závislá na ostatních krocích výpočtu je, je aktualizace horní meze účelové funkce m_{min} popsaná v podkapitole 4.2.2. V dnešní době je již mnoho moderních počítačů vybaveno jedním nebo více procesory s několika jádry. Zvýšení počítačového výkonu se též může získat zapojením několika počítačů do společné sítě, tzv. clusteru. V této variantě je možné i ze starších jednoprocesorových počítačů s jedním jádrem získat levný ale efektivní nástroj k řešení úloh.

5.1 Paralelizace v programu MATLAB

Matlab umožňuje využít nejen více jader procesoru v počítači ale umí komunikovat i s výpočetním clusterem. V rámci jednoho počítače je možné pracovat s více procesy pomocí *Parallel Computing Toolbox* [80]. Jejich maximální počet je dvanáct. V rámci clusteru je třeba mít ještě *MATLAB Distributed Computing Server* [78] naistalovaný na všech počítačích, na kterých bude počítáno (tzv. workery) a *Parallel Computing Toolbox* na počítači, na kterém se bude paralelní kód vytvářet. V této práci bude využit pouze *Parallel Computing Toolbox* kvůli analýze chování úlohy.

Nejjednodušším, ale zároveň nejméně účinným způsobem paralelizace je povolení využití více jader při výpočtu. K tomu je možné použít příkaz `maxNumCompThreads(N)`, kde `N` je počet jader, které se k výpočtu použijí [55]. Pak není nutné měnit původně sestavený kód, což je na jednu stranu výhodné z hlediska tvoření kódu, na druhou stranu uživatel nemůže paralelizaci ovlivnit. Není možné určit, které části budou paralelně počítány a jak bude nakládáno s daty mezi jednotlivými procesy. Tato varianta je tedy většinou lepší než žádná. Na obrázku 5.1 je znázornění zrychlení výpočtu při povolení použití 1, 2, 4 a 8 jader v porovnání s ideálním lineárním nárůstem rychlosti. Jak je vidět, pro metodu větví a mezí není tento způsob paralelizace účinný. Využití



Obrázek 5.1: Graf znázorňující zrychlení algoritmu při použití `maxNumCompThreads()` pro 25-prutovou konstrukci

procesoru, který má virtuálních 8 jader, bylo při `N` rovno 8 jen 15 %, což je stejné využití jako při `N` rovno jedné.

Paralelní toolbox nabízí nahrazení `for` cyklu `parfor` cyklem. Smyčky cyklu se pak počítají nezávisle na sobě na jednotlivých jádrech. Změna původního sériového kódu probíhá pouze v několika příkazech. Na začátku se otevře `N` procesů, tzv. labů, příkazem `matlabpool open N`. Počet procesů může být maximálně roven počtu jader respektive virtuálních jader procesoru anebo může být menší. Na konci algoritmu se pak laby ukončí příkazem `matlabpool close`. Matlab pak v `parfor` cyklu sám rozdělí jednotlivé na sobě nezávislé smyčky a data na konci cyklu sesbírá. Tento způsob nicméně neumožňuje využití sdílené paměti, tím pádem není možné upravovat a aktualizovat horní mez účelové funkce podle dosud nejlepšího nalezeného řešení ani mezi jednotlivými smyčkami v rámci jednoho labu ani mezi laby vzájemně. Paralelizace by pak byla na úkor metody větví a mezí, tím pádem je tato metoda nevhodná.

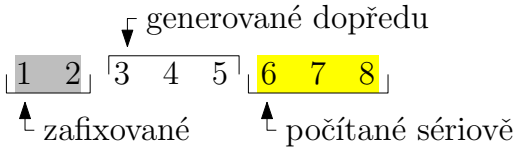
Vhodnější variantou paralelního výpočtu v *Parallel Computing Toolbox*u je metoda `spmd` (*Single Program Multiple Data*). Stejný program poběží na všech procesech, na každém procesu budou ale k dispozici jedinečná data pro výpočet. Opět je třeba otevřít potřebný počet labů pomocí `matlabpool open N`. Data se pak na jednotlivé procesy pošlou příkazem `labSend(data,X)`, kde `X` je číslo labu, na který se posílají. Poté je potřeba data přijmout příkazem `labReceive(data,Y)`, kde `Y` je číslo labu, od kterého se data přijímají. V průběhu výpočtu respektive na jeho konci se data sbírají např. příkazem `gcat()` do jednoho vektoru respektive jedné matice. V následujícím kódu je ukázáno užití `spmd` metody na jednoduchém případě. Vygeneruje se matice náhodných čísel a cílem je najít minimální hodnotu součtu řádku této matice.

```
mat = rand(3*8,5)*10; % matice náhodných čísel

matlabpool open 8     % otevře se 8 labů
spmd
   if labindex == 1   % master
      % poslání na slaves
      for ii=2:numlabs
         labSend( mat((ii-1)*3+1 : ii*3,:), ii);
      end
      % ponechání dat na masteru (stejný počet jako byl poslán na slaves)
      par_data = mat(1:3,:);

   % přijetí dat
   elseif labindex ~= 1  % slaves
      par_data = labReceive(1);   % od mastera
   end

   loc_res = min(sum(par_data,2)) % minimum sumy řádků na jednotlivých procesech
   loc_res_all = gcat(loc_res)    % sesbírání dat z procesů do vektoru
   res = min(loc_res_all)         % výsledek

end
matlabpool close
```

5.1.1 Paralelní verze modifikované metody větví a mezí

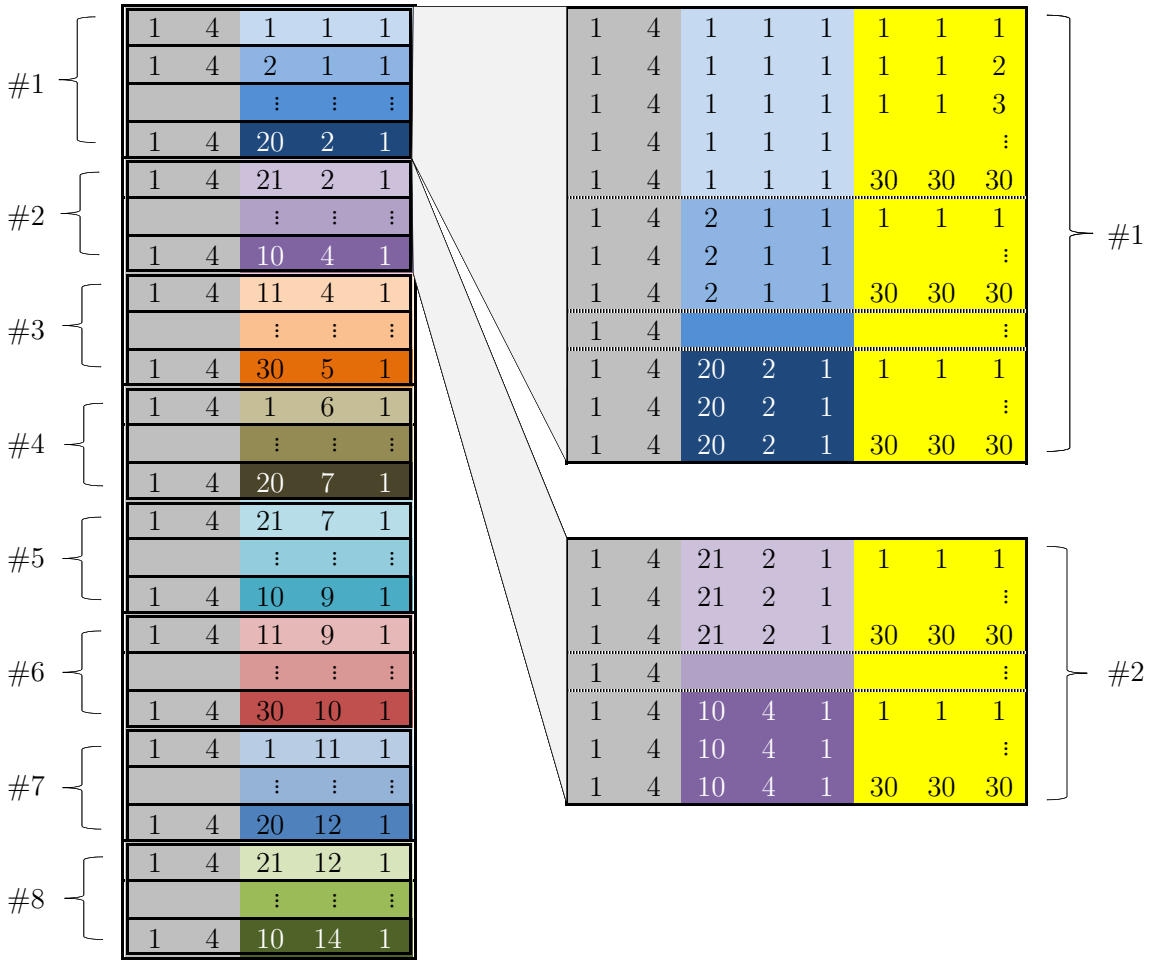
V paralelní verzi dochází k drobným úpravám kódu, algoritmus popsaný v podkapitole 4.2.2 ale zůstává zachován.

Nejprve se dopředu vygenerují kombinace průřezů na zvolených prutech, které se následně rozdělí na jednotlivé procesy. Má-li například 25-prutová konstrukce 8 skupin, je možné např. 3 skupiny určit jako generované předem a zbytek dopočítat sériově na jednotlivých labech. Tím se v podstatě změní jen pořadí výpočtu jednotlivých větví. Pro testování metody je ještě možné některé skupiny zafixovat na určitých plochách viz obrázek 5.2.



Obrázek 5.2: Rozdělení proměnných (skupin prutů) dle způsobu jejich generování pro 8 skupin (nebo prutů)

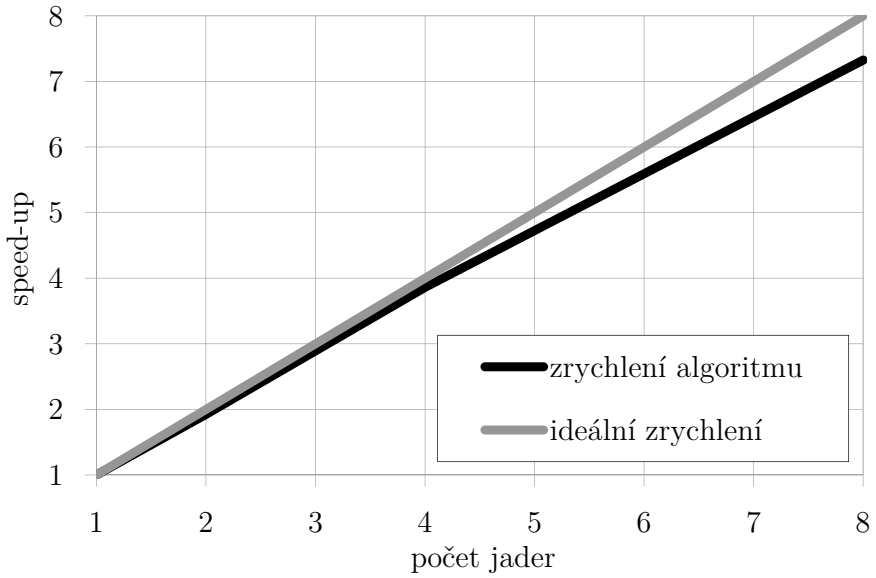
Vygenerované kombinace se posílají ve smyčkách v předem určeném počtu na jednotlivé procesy a jsou použity do výpočtu metody větví a mezí. Na obrázku 5.3 jsou vypsané hodnoty proměnných v první smyčce, pokud se předem vygenerované kombinace posílají po padesáti. Šedé proměnné jsou zafixované na stejných hodnotách, barevné proměnné jsou již vygenerovány a žluté proměnné se generují sériově v metodě



Obrázek 5.3: Hodnoty proměnných v první smyčce, posílají-li se předem vygenerované kombinace po padesáti

větví a mezí. V první tabulce jsou stručně vypsány hodnoty pro zafixované proměnné a vygenerované hodnoty dopředu. V tabulkách proces #1 a proces #2 jsou rozepsány hodnoty včetně sériového vyčíslení metodou větví a mezí. Intuitivně se stejný postup provede i na ostatních procesech.

Po každé smyčce se sesbírají nejlepší hodnoty horní meze m_{max}, vybere se nejmenší z nich a ta se poskytne se k dispozici jako nová hodnota horní meze. Je nutné odhadnout, po jakém množství dat se budou kombinace posílat a tedy jak často bude aktualizována horní mez. Jedná se o synchronizované posílání dat, výpočet tedy bude v každé smyčce trvat stejně dlouho jako výpočet na nejvíce vytíženém procesu. Pokud se sesbíraná data budou posílat často, bude čas strávený na komunikaci dlouhý. Naproti tomu pokud se budou data posílat méně často, budou se omezující podmínky pro řešení mezi horní a dolní mezí počítat i pro zadání kombinací, které by se při častější aktualizaci horní meze nepočítaly. Je tedy nutné odhadnout vhodnou hranici mezi těmito dvěma aspekty. Výpočet končí, když jsou využity všechny předem vygenerované kombinace.



Obrázek 5.4: Graf znázorňující zrychlení algoritmu při použití metody `spmd` pro 25-prutovou konstrukci

Pro paralelní způsob výpočtu je důležité, jak dobře je úloha škálovatelná. Ideální stav je, pokud je na n procesech dosaženo n-násobné zrychlení. V praxi je však tohoto stavu těžké dosáhnout, protože se čas ztrácí na komunikaci mezi procesy. Na obrázku 5.4 je znázorněno zrychlení úlohy na jednom až osmi procesech pro 25-prutovou konstrukci s rozdělením proměnných dle obrázku 5.2. Pro výpočty byl použit počítač HP Xeon Z600 Workstation s dvěma čtyřjádrovými procesory Intel Xeon E5520 s frekvencí 2,27 GHz. Výpočet probíhal na systému Debian GNU/Linux v Matlabu R2009a 64-bitové verzi.

5.2 Paralelizace v jazyce C/C++ s využitím MPI

MPI (*Message Passing Interface*) je protokol implementovaný v několika jazycích jako např. C/C++, Fortran, Python apod. a slouží ke komunikaci mezi jednotlivými procesy. Komunikace probíhá na úrovni posílání a přijímání zpráv. MPI bylo vyvinuto čtyřiceti organizacemi, které se později sdružily pod MPI fórum, a bylo poprvé představeno na konferenci Supercomputing 93 v listopadu roku 1993. Nyní se používá jeho druhá verze MPI-2 (první byla MPI-1) [22].

Na začátku každého programu musí být inicializace MPI funkcí `MPI_Init(&argc, &argv)`, na konci pak ukončení funkcí `MPI_Finalize()`. Jednotlivým procesům jsou po zavolání funkce `MPI_Comm_rank(MPI_COMM_WORLD, &rank)` přiřazena pořadová čísla do proměnné `rank` od 0 do $N-1$, kde N je počet všech procesů. Celkový počet procesů N je dostupný v proměnné `size` po zavolání funkce `MPI_Comm_size(MPI_COMM_WORLD, &size)`. `MPI_COMM_WORLD` je komunikátor umožňující pracovat se všemi procesy, které

jsou používané MPI. Pro ukázku je uveden kód, který inicializuje MPI a z každého procesu vypíše své číslo a jméno počítače, na kterém proces běží.

```
#include<stdio.h>
#include<mpi.h>      // knihovna MPI pro jazyk C/C++
#include <unistd.h>  // knihovna MPI obsahující funkci gethostname

int main(int argc, char *argv[])
{
     int ierror, rank, size;
     char uname[256];

     MPI_Init(&argc, &argv);  // inicializace MPI

     gethostname(uname, 255); // získání jména počítače, na kterém proces běží
     MPI_Comm_rank(MPI_COMM_WORLD, &rank); // získání pořadového čísla procesu
     MPI_Comm_size(MPI_COMM_WORLD, &size); // získání celkového počtu procesů

     printf("Process #%d of %d is working on %s ... \n", rank, size,uname);

     MPI_Finalize(); // ukončení MPI
     return 0;
}
```

K posílání dat je vhodné využít jeden řídící proces (většinou na počítači, na kterém je hlavní program spuštěn). Slouží k němu funkce `int MPI_Send(void* buf, int count, MPI_Datatype datatype, int dest, int tag, MPI_Comm comm)`, která existuje i v jiných alternativách pro synchronní, asynchronní, blokující nebo neblokující posílání, detaily a popis viz [22]. Pro synchronní blokující posílání slouží funkce `int MPI_Ssend()` se stejnými argumenty jako má funkce `int MPI_Send()`. Synchronní posílání znamená poslání dat ve stejnou dobu, procesy na sebe tedy počkají, dokud není dokončen výpočet i na posledním procesu, a blokující znamená, že program nepoběží dál, dokud nejsou všechna data bezpečně předána. Pro synchronní blokované přijímání dat slouží funkce `int MPI_Recv(void* buf, int count, MPI_Datatype datatype, int source, int tag, MPI_Comm comm, MPI_Status *status)`. Pokud je odesíláno více dat, je vhodné je uspořádat do jednorozměrného pole. Dynamicky alokované dvou a vícerozměrnéá pole totiž nemají data v paměti počítače uspořádaná za sebou a je tedy nutné odeslat toto pole po jednotlivých jednorozměrných polích po řádcích. Funkcím `MPI_Send` a `MPI_Recv` je totiž předáván pouze ukazatel na začátek tohoto pole `buf` a množství dat, které má být posláno, v proměnné `count`. Dále je potřeba definovat, jaký typ proměnných je posílán, tedy jedná-li se např. o typ double nebo int. Tyto typy mají svou vlastní definici, např. `MPI_INT` nebo `MPI_DOUBLE`. Proměnná `dest` obsahuje číslo procesu, na který se data odesílají a proměnná `source` číslo procesu, od kterého se data přijímají. Proměnná `tag` obsahuje číslo zprávy, které může být libovolné, ale stejné pro odeslání i příjem a `comm` je typ komunikátoru. Nejčastěji používaný je již zmíněný `MPI_COMM_WORLD`. Protokol MPI se pak sám postará o správnou distribuci dat.

Na následujících řádcích je stejný typ programu, který byl sestaven pomocí metody `spmd` v prostředí MATLAB.

```
#include<stdio.h>
#include<mpi.h>
#include<stdlib.h>
#include<math.h>

typedef struct {
  double value;
  int myrank;
} doubleint;  // definování struktury pro sesbírání dat

int main(int argc, char *argv[])
{
  int ierror, rank, size;
  doubleint Lbest, Gbest;

  MPI_Init(&argc, &argv);
  MPI_Comm_rank(MPI_COMM_WORLD, &rank);
  MPI_Comm_size(MPI_COMM_WORLD, &size);

  MPI_Status status ;
  double *mat, *par_data, *loc_res_temp;
  int i, ii, j, start;
  par_data = new double[3*5];
  loc_res_temp = new double[3];
  Lbest.myrank = rank;

  // řídící proces
  if( rank == 0) {
     mat = new double[3*8*5];
     for (i=0; i<3*8*5; i++)
        mat[i] = rand()/10E5;

     // co se ponechá řídícímu procesu
     for(i=0; i<3*5; i++)
        par_data[i] = mat[i];

     // poslání dat podřízeným procesům
     for(ii = 1; ii < size; ii++) {
        start = ii*3*5;
        MPI_Ssend(&mat[start], 3*5, MPI_DOUBLE, ii, 101, MPI_COMM_WORLD);
     }
  }

  // přijetí dat podřízenými procesy
  else
     MPI_Recv(&par_data[0], 3*5, MPI_DOUBLE, 0, 101, MPI_COMM_WORLD, &status);

  // suma řádků matice par_data (lokální mat)
  for (i=0; i<3; i++) {
     for (j=0; j<5; j++)
        loc_res_temp[i] = loc_res_temp[i] + par_data[i*5+j];
  }

  // nejlepší nalezené řešení na procesu
  Lbest.value = loc_res_temp[0];
```

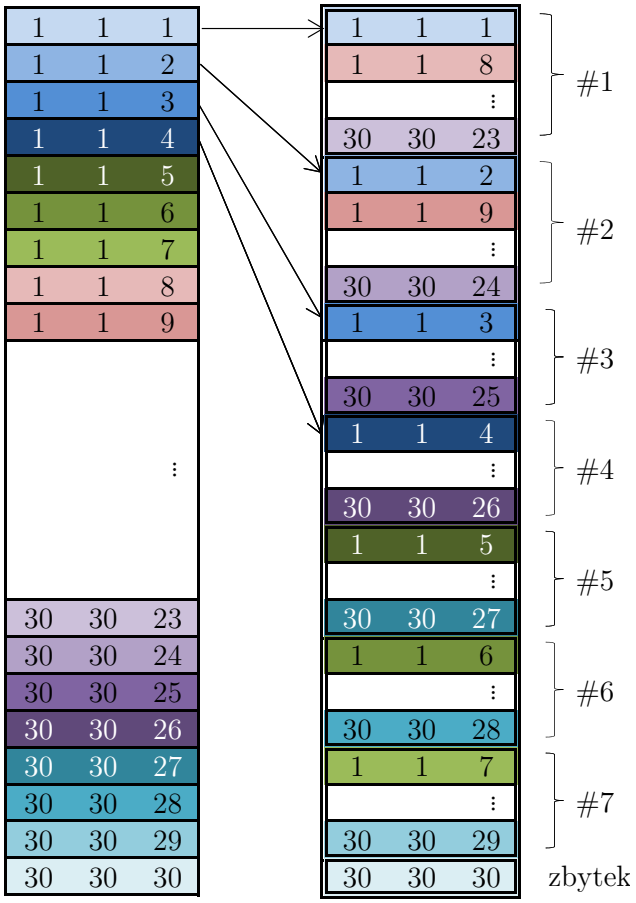
```
56    for(i=0; i<3; i++) {
57       if (Lbest.value > loc_res_temp[i]) Lbest.value = loc_res_temp[i];
58    }
59
60    // sesbírání nejlepších nalezených řešení na procesech procesem #0
61    MPI_Reduce(&Lbest, &Gbest, 1, MPI_DOUBLE_INT, MPI_MINLOC, 0, MPI_COMM_WORLD);
62
63    // tisk řešení
64    if(rank == 0)
65       printf("res = %lf na procesu #%d\n", Gbest.value, Gbest.myrank);
66
67    MPI_Finalize();
68  //  scanf("%*c");
69    return 0;
70  }
```

Sesbírání nejlepších nalezených řešení je uskutečněno funkcí `int MPI_Reduce(void* sendbuf, void* recvbuf, int count, MPI_Datatype datatype, MPI_Op op, int root, MPI_Comm comm)`, kde `sendbuf` je proměnná, která se sbírá, `recvbuf` je proměnná, do které se data ukládají, `count` je počet očekávaných dat a `datatype` je typ proměnné. Proměnná `op` určuje operaci, která se pro sbírání uskuteční. Pokud je třeba získat minimum ze sesbíraných hodnot, využije se `MPI_MIN`, pro maximum z hodnot se využije `MPI_MAX`. Pokud je ale třeba získat i proces, na kterém je nejlepší hodnota nalezena, je třeba použít `MPI_MINLOC`. S touto operací je ale potřeba definovat strukturu, do které se bude ukládat nejlepší hodnota a číslo procesu. Tato struktura se pak předává do funkce `MPI_Reduce`. Proměnná `root` je číslo procesu, kam se sbírá nejlepší hodnota ze všech procesů.

5.2.1 Paralelní verze modifikované metody větví a mezí

Paralelní verze metody větví a mezí v jazyce C/C++ s využitím MPI je založena na stejném principu jako v prostředí MATLAB pomocí procedury `spmd`. Rozdíly implementace jsou v posílání dat. Předem generované kombinace se rozdělí a odešlou najednou. Po určitém počtu spočítaných kombinací se sbírá nejlepší řešení m_{max}, číslo procesu a odpovídající kombinace pořadových čísel ploch příčných průřezů. Po vyčerpání předem vygenerovaných dat úloha končí. Rozdíl je též v tom, že řídícímu procesu zabírá čas komunikace, proto je pro něj ponechána menší část dat než pro procesy podřízené.

Předem generované kombinace jsou uspořádány od nejmenší po největší. Pokud je k dispozici N procesů, na které se budou data posílat, a kombinace se rozdělí na N částí, bude úloha nevyvážená a nárůst zrychlení úlohy nebude ideální. Na prvním procesu totiž budou úlohy, které spadají převážně pod dolní mez, naopak na posledním procesu budou úlohy, které budou nad horní mezí případně mezi dolní a horní mezí a bude tedy nutné počítat omezující podmínky. Proto je vhodné data před odesláním přeuspořádat, aby bylo využití všech procesů srovnatelné. Toho je možné dosáhnout několika způsoby. Jednou z variant je postupné rozložení kombinací vzhledem k jednotlivým procesům.



Obrázek 5.5: Dopředu vygenerované kombinace a jejich přeuspořádání s ohledem na jednotlivé procesy

Jelikož je předem známý počet vygenerovaných kombinací a počet procesů, dá se snadno zjistit, kde bude ležet počátek odesílaného balíku dat řídícím procesem na proces podřízený. Kombinace se přeuspořádají tak, že prvních N vygenerovaných dat se postupně přemístí na počátek odesílaných balíků a takto se pokračuje až do konce viz obrázek 5.5. Bylo záměrně zvoleno pouze sedm procesů, aby bylo vidět, že ne vždy jsou data počtem procesů dělitelná. Data, která zbydou, se zařadí na konec.

Kapitola 6

Výsledky

6.1 Rešerše publikovaných výsledků

Konstrukce popsané v kapitole 2 jsou v literatuře hojně využívány nejen pro rozměrovou optimalizaci. Optima těchto konstrukcí jsou získávána pomocí různých metod již přes čtyřicet let. Často jsou ale používány metody (např. heuristické), při kterých není jistota, je-li obdržené optimum globální. Pro praktické účely slouží dobře i optima lokální. Nicméně je vhodné získané výsledky porovnávat s nejlepším možným řešením, aby se zjistila kvalita použité metody. Tím absolutně nejlepším je optimum globální, které je výpočetně náročné získat, ale pokud je již jednou spočítané, poslouží jako etalon kvality při vývoji nových metod.

V následujících tabulkách jsou uvedeny publikované výsledky jak pro diskrétní, tak pro spojitou optimalizaci. Přehled použitých metod je jednak v následujících tabulkách a jednak v příloze E včetně významu jednotlivých zkratek algoritmů. Výsledky jsou rozděleny podle toho, zda-li vyhovují omezujícím podmínkám či nikoliv. Ve všech tabulkách pro řešení splňující omezující podmínky jsou řešení seřazena podle hodnoty účelové funkce, tedy hmotnosti konstrukce. Nevyhovující řešení jsou srovnána též podle hmotnosti konstrukce, pokud jsou maximální hodnoty posunu a napětí po zaokrouhlení shora rovny limitním hodnotám v zadání. V ostatním případě, tedy pokud řešení hrubě nesplní omezující podmínky, jsou srovnána dle data publikování.

Pro všechna řešení jsou uvedeny spočítané absolutní hodnoty maximálního posunu ve vodorovném i svislém směru $\max|w_j|$, kde j je pořadové číslo volných posunů, a absolutní hodnoty maximálních napětí $\max|\sigma_i|$, kde i je pořadové číslo prutu. Pro výpočet byla použita MKP. Kurzívou jsou zvýrazněny nevyhovující hodnoty. Dále jsou pro porovnání v tabulkách uvedeny limitní hodnoty deformace w_{lim} a napětí σ_{lim} zadané v rámci optimalizační úlohy. Jsou zde též uvedeny publikované hmotnosti konstrukce m_{ref} a tyto hodnoty jsou pro kontrolu ještě přepočítány v řádku m. I zde dochází u publikovaných výsledků k odlišnostem od výsledků získaných v této práci.

Proměnná	Jednotky	Cai ES 1996 [10]	Lemonge GA 2004 [50]	Kripka SA 2004 [43]	Nosek ES 2008 [58]	Barbosa GA 2008 [6]
A_1	in^2	33,50	33,50	33,50	33,50	33,50
A_2	in^2	1,62	1,62	1,62	1,62	1,62
A_3	in^2	22,90	22,90	22,90	22,90	22,90
A_4	in^2	14,20	14,20	14,20	14,20	14,20
A_5	in^2	1,62	1,62	1,62	1,62	1,62
A_6	in^2	1,62	1,62	1,62	1,62	1,62
A_7	in^2	7,97	7,97	7,97	7,97	7,97
A_8	in^2	22,90	22,90	22,90	22,90	22,90
A_9	in^2	22,00	22,00	22,00	22,00	22,00
A_{10}	in^2	1,62	1,62	1,62	1,62	1,62
m_{ref}	lb	5490,71	5490,74	5490,74	5490,74	5490,74
m	lb	**5490,74**	**5490,74**	**5490,74**	**5490,74**	**5490,74**
$\max \lvert w_j \rvert$	in	2,00	2,00	2,00	2,00	2,00
$\max \lvert \sigma_i \rvert$	ksi	14,20	14,20	14,20	14,20	14,20
w_{lim}	in	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25

Proměnná	Jednotky	Tong GA 2000 [82]	Sousa GEO 2005 [14]	Li HPSO 2009 [53]	Rajeev GA 1992 [64]	Shih MOP, MDNLP 1997 [69]
A_1	in^2	33,50	33,50	30,00	33,50	33,50
A_2	in^2	1,62	2,13	1,62	1,62	1,62
A_3	in^2	22,90	22,00	22,90	22,00	33,50
A_4	in^2	15,50	13,90	13,50	15,50	26,50
A_5	in^2	1,62	1,62	1,62	1,62	1,62
A_6	in^2	1,62	1,99	1,62	1,80	1,62
A_7	in^2	7,97	7,97	7,97	14,20	13,90
A_8	in^2	22,00	22,90	26,50	19,90	33,50
A_9	in^2	22,00	22,90	22,00	19,90	33,50
A_{10}	in^2	1,62	1,62	1,80	2,62	1,80
m_{ref}	lb	5491,71	5525,04	5531,98	5620,08	7751,36
m	lb	5491,72	5525,04	5531,98	5620,06	7751,36
$\max \lvert w_j \rvert$	in	2,00	2,00	2,00	2,00	1,43
$\max \lvert \sigma_i \rvert$	ksi	14,07	14,36	14,87	9,45	9,01
w_{lim}	in	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25

Tabulka 6.1: Vyhovující výsledky pro 10-prutovou konstrukci - diskrétní problém

6.1.1 Desetiprutová konstrukce

V tabulkách 6.1 až 6.4 jsou uvedeny publikované výsledky pro desetiprutovou konstrukci. V tabulce 6.1 jsou uvedena diskrétní optima, která splňují omezující podmínky ve formě posunů i napětí. V tabulce 6.2 jsou uvedena diskrétní řešení, která těmto

Proměnná	Jednotky	Wu GA 1995 [88]	Wu GA 1995 [88]	Wu GA 1995 [88]	Wu GA 1995 [88]	Camp GA 1998 [12]	Camp GA 1998 [12]	Ghasemi GA 1999 [27]	Ghasemi GA 1999 [27]
A_1	in^2	26,50	26,50	26,50	26,50	26,50	26,50	33,50	33,50
A_2	in^2	1,62	1,80	1,62	1,62	13,50	19,90	1,62	1,62
A_3	in^2	16,90	16,00	16,00	16,00	1,62	2,13	22,90	22,00
A_4	in^2	11,50	13,50	14,20	13,90	1,62	1,99	14,20	13,90
A_5	in^2	1,62	1,62	1,80	1,62	30,00	30,00	1,62	1,62
A_6	in^2	1,62	1,62	1,62	1,99	1,62	1,62	1,62	1,62
A_7	in^2	4,97	4,97	5,12	4,97	22,90	22,90	7,22	7,97
A_8	in^2	16,90	16,00	16,00	16,00	7,22	7,22	22,00	22,90
A_9	in^2	16,90	19,90	18,80	18,80	1,62	1,62	22,90	22,90
A_{10}	in^2	1,99	1,80	2,38	2,62	22,00	19,90	1,62	1,62
m_{ref}	lb	4379,26	4369,84	4376,20	4376,83	5556,90	5586,10	5453,32	5447,54
m	lb	4226,52	4369,84	4376,20	4376,83	5430,95	5586,12	5452,55	5493,36
$\max \lvert w_j \rvert$	in	*2,67*	*2,60*	*2,62*	*2,62*	*8,33*	*7,58*	*2,02*	*2,00*
$\max \lvert \sigma_i \rvert$	ksi	19,98	19,97	19,33	20,02	*79,01*	*68,50*	15,22	14,32
w_{lim}	in	2	2	2	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25	25	25	25

Tabulka 6.2: Nevyhovující výsledky pro 10-prutovou konstrukci - diskrétní problém

podmínkám nevyhovují, je-li použit výpočet na základě deformační varianty metody konečných prvků. Některé metody totiž mohou při optimalizaci narušit omezující podmínky např. jejich relaxací a výsledek pak již optimem není. Tyto výsledky pak není možné porovnávat s řešeními, které omezujícím podmínkám vyhovují, i když je úloha stejná a hodnota účelové funkce nízká. Případně může během výpočtu dojít k zaokrouhlovacím chybám jednak v rámci samotného optimalizačního programu a jednak v rámci zaokrouhlovací chyby počítače. Otázkou je, zda-li tyto výsledky uvažovat za správné anebo se striktně držet omezujících podmínek. V této práci hodnoty lehce přesahující hodnoty omezujících podmínek uvažovány jako optima nebudou. V tabulce 6.3 jsou uvedena spojitá optima pro tuto konstrukci, která omezující podmínky splňují. Jako dolní mez je zde použita jednak plocha 0,1 in^2 a jednak 1,62 in^2. Řešení s dolní mezí 0,1 in^2 jsou zde uvedena jako ilustrační, neboť optimalizační úloha je jinak naprosto totožná. V tabulce 6.4 jsou uvedena řešení těmto podmínkám nevyhovující. Zde se většinou jedná (až na poslední tři případy autorů Fleuryho, Zhou a Campa) právě o případ drobného přesahu omezujících podmínek, ať už ve formě posunu, napětí nebo obou dvou. Hodnoty přesahující limitní hodnoty v zadání úlohy jsou odlišeny kurzívou.

Nejlepší řešení s hmotností 5490,74 liber je pro diskrétní optimalizaci poprvé publikováno autory Caiem a Thieraufem roku 1996. K jeho výpočtu je použit evoluční algoritmus. Toto řešení je později publikováno i dalšími autory a je získáno pomocí

Proměnná	Jednotky	Nosek DE 2008 [58]	Hadidi MABC 2010 [32]	Li PSOPC 2007 [52]	Wang TPO 1984 [30]	Renwei 1985 [65]
A_1	in^2	30,4725	30,6573	30,569	30,03	30,59
A_2	in^2	0,1002	0,1	0,1	0,10	0,1
A_3	in^2	23,1728	23,0429	22,974	23,27	23,27
A_4	in^2	15,21	15,2821	15,148	15,23	15,19
A_5	in^2	0,1001	0,1	0,1	0,10	0,1
A_6	in^2	0,5614	0,5626	0,547	0,55	0,46
A_7	in^2	7,4641	7,4721	7,493	7,47	7,5
A_8	in^2	21,0887	21,0084	21,159	21,33	21,07
A_9	in^2	21,5265	21,5094	21,556	21,53	21,48
A_{10}	in^2	0,1	0,1	0,1	0,10	0,1
m_{ref}	lb	5060,9	5060,97	5061		5062,17
m	lb	**5060,925**	5060,978	5061,033	5061,556	5062,781
$\max \lvert w_j \rvert$	in	2,000	2,000	2,000	2,000	2,000
$\max \lvert \sigma_i \rvert$	ksi	24,998	24,995	24,998	24,993	24,815
w_{lim}	in	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25

Proměnná	Jednotky	Schmit ACCESS 1976 [67]	Venkayya CA 1971 [83]	Barbosa GA 2008 [6]	Shih MOP, MDNLP 1997 [69]
A_1	in^2	30,67	30,416	22,736	33,5
A_2	in^2	0,1	0,128	0,1	1,629
A_3	in^2	23,76	23,408	22,736	33,5
A_4	in^2	14,59	14,904	22,736	25,4
A_5	in^2	0,1	0,101	0,1	1,62
A_6	in^2	0,1	0,101	0,1	1,629
A_7	in^2	8,578	8,696	22,736	13,64
A_8	in^2	21,07	21,084	22,736	33,5
A_9	in^2	20,96	21,077	22,736	33,5
A_{10}	in^2	0,1	0,186	0,1	1,83
m_{ref}	lb	5076,85	5084,9	5943,847	7701,3
m	lb	5077,150	5084,773	5943,964	7700,695
$\max \lvert w_j \rvert$	in	2,000	2,000	2,000	1,436
$\max \lvert \sigma_i \rvert$	ksi	20,349	20,001	8,811	9,149
w_{lim}	in	2	2	2	2
σ_{lim}	ksi	25	25	25	25

Tabulka 6.3: Vyhovující výsledky pro 10-prutovou konstrukci - spojitý problém

různých optimalizačních metod jako je genetický algoritmus autorů Lemonge a Barbosy nebo simulované žíhání autora Kripky. Pro spojitou úlohu s dolní mezí 0,1 in^2 je nejlepší řešení publikováno autorem Noskem roku 2008 s hmotností 5060,93 liber. Velmi blízká řešení jsou též získána metodou včelího roje (MABC) autorem Hadidi

Proměnná	Jednotky	Ghasemi GA 1999 [27]	Fleury 1980 [21]	Lemonge GA 2004 [50]	Lee HS 2004 [48]	Lee 2009 [47]	Kaveh HPSACO 2009 [41]
A_1	in^2	25,73	30,95	29,22568	30,15	30,15	30,307
A_2	in^2	0,109	0,1	0,1	0,102	0,102	0,1
A_3	in^2	24,85	26,08	24,18212	22,71	22,71	23,434
A_4	in^2	16,35	15,04	14,94714	15,27	15,27	15,505
A_5	in^2	0,106	0,1	0,1	0,102	0,102	0,1
A_6	in^2	0,109	0,196	0,39463	0,544	0,544	0,5241
A_7	in^2	8,7	8,182	7,49579	7,541	7,541	7,4365
A_8	in^2	21,41	20,22	21,92486	21,56	21,56	21,079
A_9	in^2	22,3	20,22	21,29088	21,45	21,45	21,229
A_{10}	in^2	0,122	0,1	0,1	0,1	0,1	0,1
m_{ref}	lb	5095,65	5089,80	5069,09	5057,88	5057,88	5056,56
m	lb	5095,64	5089,30	5069,09	5058,34	5058,34	5056,59
$\max \lvert w_j \rvert$	in	*2,01*	*2,00*	*2,00*	*2,00*	*2,00*	*2,00*
$\max \lvert \sigma_i \rvert$	ksi	18,53	20,64	24,75	25,00	25,00	25,00
w_{lim}	in	2	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25	25

Proměnná	Jednotky	Fleury 1980 [21]	Perez PSO 2007 [60]	Terai 1974 [74]	Fleury HOC 1979 [20]	Zhou DCOC 1993 [91]	Camp 1998 [12]
A_1	in^2	30,52	33,5	31,03	23,2	30,73	24,07
A_2	in^2	0,1	0,1	0,1	15,223	23,941	13,96
A_3	in^2	23,2	22,76	22,5	0,551	0,1	0,56
A_4	in^2	15,22	14,42	15,32	0,1	14,733	0,1
A_5	in^2	0,1	0,1	0,1	30,522	8,541	28,92
A_6	in^2	0,551	0,1	0,79	0,1	20,951	0,1
A_7	in^2	7,457	7,534	5,8	21,036	20,836	21,95
A_8	in^2	21,04	20,46	21,93	7,457	0,1	7,69
A_9	in^2	21,53	20,4	21,67	0,1	0,1	0,1
A_{10}	in^2	0,1	0,1	0,1	21,528	0,1	22,9
m_{ref}	lb	5060,85	5024,10	5034,40	5060,85	5076,67	5077,00
m	lb	5060,93	5024,19	5034,37	5060,80	4639,93	5117,55
$\max \lvert w_j \rvert$	in	2,00	*2,04*	2,00	*37,63*	*211,24*	*37,41*
$\max \lvert \sigma_i \rvert$	ksi	*25,00*	*25,02*	*35,57*	*788,22*	*2214,47*	*788,16*
w_{lim}	in	2	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25	25

Tabulka 6.4: Nevyhovující výsledky pro 10-prutovou konstrukci - spojitý problém

s hmotností 5060,98 liber nebo optimalizací hejnem částic autorem Li s hmotností 5061,03 liber. Řešení s dolní mezí 1,62 in^2 je publikováno autorem Shihem s hmotností 7700,70 liber. Při porovnání s nejlepším nalezeným diskrétním řešením s hmotností 5490,71 liber je však toto řešení výrazně horší.

Proměnná	Jednotky	Kripka SA 2004 [43]	Nosek ES, DE 2008 [58]	Lemonge GA 2003 [50]	Li HPSO 2009 [53]	Hasançebi PSO, HS, SA 2009 [34]
A_1	in^2	0,1	0,1	0,1	0,1	0,1
A_2	in^2	0,4	0,4	0,3	0,3	0,3
A_3	in^2	3,4	3,4	3,4	3,4	3,4
A_4	in^2	0,1	0,1	0,1	0,1	0,1
A_5	in^2	2,2	2,2	2,1	2,1	2,1
A_6	in^2	1,0	1,0	1,0	1,0	1,0
A_7	in^2	0,4	0,4	0,5	0,5	0,5
A_8	in^2	3,4	3,4	3,4	3,4	3,4
m_{ref}	lb	484,33	484,33	484,85	484,85	484,85
m	lb	**484,33**	**484,33**	484,85	484,85	484,85
$\max \lvert\sigma_i\rvert$	in	0,350	0,350	0,350	0,350	0,350
$\max \lvert w_j\rvert$	ksi	6,196	6,196	6,111	6,111	6,111
σ_{lim}	in	0,35	0,35	0,35	0,35	0,35
w_{lim}	ksi	40	40	40	40	40

Proměnná	Jednotky	Lee HS 2005 [49]	Tong 2001 [82]	Hasançebi ES 2009 [34]	Hasançebi AC 2009 [34]	Hasançebi SGA 2009 [34]
A_1	in^2	0,1	0,1	0,1	0,1	0,1
A_2	in^2	0,3	0,5	0,5	0,5	0,2
A_3	in^2	3,4	3,4	3,4	3,4	3,4
A_4	in^2	0,1	0,1	0,1	0,1	0,1
A_5	in^2	2,1	1,9	1,9	1,9	2,0
A_6	in^2	1,0	1,0	0,9	1,0	1,0
A_7	in^2	0,5	0,4	0,5	0,4	0,6
A_8	in^2	3,4	3,4	3,4	3,4	3,4
m_{ref}	lb	484,85	485,05	485,05	485,05	485,38
m	lb	484,85	485,05	485,05	485,05	485,38
$\max \lvert\sigma_i\rvert$	in	0,350	0,350	0,350	0,350	0,350
$\max \lvert w_j\rvert$	ksi	6,111	6,186	6,113	6,186	6,837
σ_{lim}	in	0,35	0,35	0,35	0,35	0,35
w_{lim}	ksi	40	40	40	40	40

Tabulka 6.5: Vyhovující výsledky pro 25-prutovou konstrukci - diskrétní problém

6.1.2 Dvacetipětiprutová konstrukce

V tabulkách 6.5 a 6.6 jsou uvedeny výsledky pro diskrétní respektive spojitou optimalizaci pro 25-prutovou konstrukci. Nejlepší dosud nalezené řešení bylo poprvé publikováno roku 2004 autorem Kripkou s hmotností 484,33 liber. K jeho získání byla použita metoda simulovaného žíhání (SA). V roce 2008 autor Nosek obdržel naprosto totožné řešení pomocí evolučních algoritmů a diferenciální evoluce. Je zajímavé si

Proměnná	Jednotky	Hasançebi TS 2009 [34]	Wu GA 1995 [88]	Wu GA 1995 [87]	Erbatur GA 200 [18]	Coello GA 1994 [13]	Rajeev GA 1992 [64]	Perez PSO 2007 [60]
A_1	in^2	0,1	0,1	0,1	0,1	1,5	0,1	0,1
A_2	in^2	0,4	0,5	0,6	1,2	0,7	1,8	0,4565
A_3	in^2	3,4	3,4	3,2	3,2	3,4	2,3	3,4
A_4	in^2	0,1	0,1	0,2	0,1	0,7	0,2	0,1
A_5	in^2	1,8	1,5	1,5	1,1	0,4	0,1	1,9369
A_6	in^2	0,9	0,9	1,0	0,9	0,7	0,8	0,9647
A_7	in^2	0,6	0,6	0,6	0,4	1,5	1,8	0,4423
A_8	in^2	3,4	3,4	3,4	3,4	3,2	3,0	3,4
m_{ref}	lb	485,57	486,29	491,72	493,80	539,78	546,01	483,84
m	lb	485,57	486,29	491,72	493,80	539,78	546,01	**483,84**
$\max \lvert w_j \rvert$	in	0,349	0,349	0,349	0,350	0,345	0,348	0,35
$\max \lvert \sigma_i \rvert$	ksi	6,027	6,008	6,014	6,432	6,665	6,773	6,15
w_{lim}	in	0,35	0,35	0,35	0,35	0,35	0,35	0,35
σ_{lim}	ksi	40	40	40	40	40	40	40

Tabulka 6.6: Vyhovující výsledky pro 25-prutovou konstrukci - diskrétní problém (pokračování), spojité řešení (oddělené čarou)

všimnout rozložení ploch na jednotlivých skupinách prutů. Pruty, které tvoří jakýsi obal konstrukce a směřují co nejkratší cestou z horních uzlů do podpor, mají převážně největší možné plochy ze sady, zatímco pruty, které jsou vodorovné a obal konstrukce spojují, mají nejmenší možné plochy.

Spojité optimum s hmotností 483,84 liber bylo publikováno roku 2007 autorem Perezem. Při srovnání diskrétního a spojitého optima je spojité optimum nepatrně lepší, což pozitivně vypovídá o jeho kvalitě. Tato úloha je pro spojitou optimalizaci hojně užívaná, nicméně nemá stejné zadání jako úloha diskrétní (zadání např. viz kapitola 2), neboť jsou používány odlišné zatěžovací stavy a odlišné omezení napětí v tahu a tlaku. Proto je zde uvedeno jediné spojité optimum se zadáním totožným jako u diskrétní optimalizace.

6.1.3 Padesátidvouprutová konstrukce

Výsledky pro 52-prutovou konstrukci jsou uvedeny v tabulce 6.7. Tato konstrukce je relativně nová oproti ostatním konstrukcím a nemá tedy tolik nalezených optim jako konstrukce ostatní. Tabulka je vizuálně rozdělena do dvou částí, výsledky na levé straně jsou nalezená vyhovující optima omezující podmínce, zatímco výsledek autora Kaveha omezující podmínku nesplňuje. Ta je v této úloze jediná, a to ve formě omezení maximálního napětí v tahu i tlaku. Nejlepší řešení je publikováno autorem Lemongem roku 2004 s hmotností 1903,37 kg; k jeho získání byl použit genetický algoritmus. V roce 2006 je publikováno další optimum autorem Gigerem se stejnou hmotností,

Proměnná	Jednotky	Lemonge 2004 GA [50]	Giger 2006 [28]	Li 2009 HPSO [53]	Wu 1995 GA [88]	Kaveh 2009 DHPSACO [40]
A_1	in^2	4658,055	4658,055	4658,055	4658,055	4658,055
A_2	in^2	1161,288	1161,288	1161,288	1161,288	1161,288
A_3	in^2	494,193	494,193	363,225	645,16	494,193
A_4	in^2	3303,219	3303,219	3303,219	3303,219	3303,219
A_5	in^2	940,000	940,000	940,000	1045,159	1008,385
A_6	in^2	641,289	494,193	494,193	494,193	285,161
A_7	in^2	2238,705	2238,705	2238,705	2477,414	2290,318
A_8	in^2	1008,385	1008,385	1008,385	1045,159	1008,385
A_9	in^2	363,225	363,225	388,386	285,161	388,386
A_{10}	in^2	1283,868	1283,868	1283,868	1696,771	1283,868
A_{11}	in^2	1161,288	1161,288	1161,288	1045,159	1161,288
A_{12}	in^2	494,193	641,289	792,256	641,289	506,451
m_{ref}	kg	1903,37	1903,37	1905,50	1970,14	1904,83
m	kg	**1903,37**	**1903,37**	1905,50	1972,65	1904,83
$\max \lvert\sigma_i\rvert$	MPa	179,90	179,78	179,97	178,92	*180,49*
σ_{lim}	MPa	180	180	180	180	180

Tabulka 6.7: Výsledky pro 52-prutovou konstrukci - diskrétní problém

a to 1903,37 kg. Rozdíl v těchto dvou řešeních je prohození hodnot ploch příčných řezů u skupin prutů A_6 a A_{12}. Tyto skupiny jsou použity pro vodorovné pruty na prostředním respektive horním patře konstrukce a jelikož je konstrukce pravidelná, mají i stejnou délku. Poměrné dobré je i další řešení autora Li s hmotností 1905,50 kg pomocí optimalizace hejnem částic.

6.1.4 Sedmdesátidvouprutová konstrukce

Výsledky publikované v literatuře pro 72-prutovou konstrukci jsou vypsány v tabulkách 6.8 pro diskrétní optimalizaci a 6.9 pro spojitou optimalizaci. Tabulka 6.8 je rozdělena do dvou částí, levá část obsahuje řešení s vyhovujícími omezujícími podmínkami, pravá část obsahuje řešení, která omezujícím podmínkám nevyhovují. Nejlepší diskrétní řešení bylo publikováno roku 1995 autory Wu a Chow s hmotností 400,66 liber a bylo získáno genetickým algoritmem. Tabulka 6.9 je rozdělena do třech částí. První část obsahuje optima, která vyhovují omezujícím podmínkám. Druhá a třetí část obsahuje řešení nesplňující tyto podmínky. V první a druhé části je použita dolní mez 0,1 in^2, ve třetí části je využita hodnota dolní meze 0,01 in^2. Nejlepší řešení bylo publikováno autorem Fleurym roku 1980 s hmotností 379,67 liber. Při porovnání s hodnotou diskrétního optima 400,66 liber je spojité optimum výrazně lepší. Dá se tedy uvažovat o možné rezervě k nalezení globálního optima pro diskrétní úlohu.

Proměnná	Jednotky	Wu GA 1995 [88]	Li PSO 2009 [53]	Lee HS 2005 [49]	Kaveh DHPSACO 2009 [40]
A_1	in^2	0,2	2,6	1,6	1,9
A_2	in^2	0,5	1,5	0,5	0,5
A_3	in^2	0,5	0,3	0,1	0,1
A_4	in^2	0,7	0,1	0,1	0,1
A_5	in^2	0,5	2,1	1,3	1,3
A_6	in^2	0,5	1,5	0,6	0,5
A_7	in^2	0,1	0,6	0,1	0,1
A_8	in^2	0,2	0,3	0,1	0,1
A_9	in^2	1,3	2,2	0,6	0,6
A_{10}	in^2	0,5	1,9	0,6	0,5
A_{11}	in^2	0,2	0,2	0,1	0,1
A_{12}	in^2	0,1	0,9	0,1	0,1
A_{13}	in^2	1,5	0,4	0,2	0,2
A_{14}	in^2	0,7	1,9	0,5	0,6
A_{15}	in^2	0,1	0,7	0,4	0,4
A_{16}	in^2	0,1	1,6	0,6	0,6
m_{ref}	lb	400,66	1089,88	389,08	385,54
m	lb	**400,66**	1089,88	389,08	385,54
$\max \lvert w_j \rvert$	ksi	0,25	*0,25*	*0,61*	*0,61*
$\max \lvert \sigma_i \rvert$	ksi	20,38	13,53	*30,01*	*29,40*
w_{lim}	ksi	0,25	0,25	0,25	0,25
σ_{lim}	ksi	25	25	25	25

Tabulka 6.8: Výsledky pro 72-prutovou konstrukci - diskrétní problém

Též je zajímavé si všimnout, že hodnoty u spojitého optima leží až na výjimky blízko dolní meze. U nejlepšího nalezeného řešení je tato mez dosažena na šesti skupinách prutů. Tyto skupiny sdružují vodorovné pruty druhého nejvyššího až nejnižšího patra.

Proměnná	Jednotky	Fleury	Perez	Nosek	Barbosa	Renwei	Lemonge	Erbatur	Xicheng	Zhou	Lee	Adeli	Lee	Lamberti	Sarma
				DE	GA-CPM	GA	GA		DCOC	HS	GGP	HS	SA	GA	
		1980	2007	2008	2003	1985	2004	2000	1992	1993	2004	1986	2009	2008	2000
		[21]	[60]	[58]	[5]	[65]	[50]	[18]	[89]	[91]	[48]	[1]	[47]	[44]	[66]
A_1	in^2	0,157	0,156	0,157	0,153	0,164	0,155	0,156	0,157	0,156	1,79	2,026	1,79	0,167	2,141
A_2	in^2	0,536	0,571	0,550	0,577	0,555	0,545	0,555	0,537	0,546	0,521	0,533	0,521	0,536	0,510
A_3	in^2	0,410	0,457	0,409	0,294	0,419	0,275	0,417	0,411	0,410	0,1	0,1	0,1	0,446	0,054
A_4	in^2	0,568	0,490	0,576	0,726	0,576	0,519	0,516	0,571	0,570	0,1	0,1	0,1	0,576	0,01
A_5	in^2	0,507	0,513	0,518	0,655	0,533	0,604	0,519	0,509	0,524	1,229	1,157	1,229	0,521	1,489
A_6	in^2	0,520	0,532	0,519	0,573	0,526	0,666	0,522	0,522	0,517	0,522	0,569	0,522	0,518	0,551
A_7	in^2	0,1	0,1	0,1	0,1	0,1	0,102	0,1	0,1	0,1	0,1	0,1	0,1	0,01	0,057
A_8	in^2	0,1	0,1	0,100	0,103	0,1	0,130	0,1	0,1	0,1	0,1	0,1	0,1	0,114	0,013
A_9	in^2	1,280	1,294	1,288	1,316	1,289	1,200	1,328	1,286	1,268	0,517	0,514	0,517	1,290	0,565
A_{10}	in^2	0,515	0,543	0,511	0,483	0,520	0,474	0,500	0,516	0,512	0,504	0,479	0,504	0,517	0,527
A_{11}	in^2	0,1	0,1	0,1	0,102	0,1	0,101	0,1	0,1	0,1	0,1	0,1	0,1	0,01	0,01
A_{12}	in^2	0,1	0,1	0,1	0,100	0,1	0,109	0,1	0,1	0,1	0,101	0,1	0,101	0,01	0,066
A_{13}	in^2	1,897	1,829	1,884	1,862	1,917	1,953	1,899	1,905	1,886	0,156	0,158	0,156	1,887	0,174
A_{14}	in^2	0,516	0,468	0,516	0,462	0,521	0,517	0,511	0,518	0,512	0,547	0,550	0,547	0,517	0,425
A_{15}	in^2	0,1	0,100	0,100	0,1	0,1	0,1	0,1	0,1	0,1	0,442	0,345	0,442	0,01	0,437
A_{16}	in^2	0,1	0,1	0,1	0,1	0,1	0,10105	0,1	0,1	0,1	0,59	0,498	0,59	0,01	0,641
m_{ref}	lb	379,67	381,03	379,68	384,13	379,66	387,04	379,88	380,84	379,61	379,27	379,31	379,27	165,028 kg!	372,40
m	lb	**379,67**	381,03	381,08	384,13	385,62	387,03	379,87	380,90	379,61	379,22	379,31	379,22	363,82	372,40
$\max \lvert w_j \rvert$	ksi	0,25	0,25	0,25	0,25	0,25	*0,25*	*0,25*	0,25	*0,25*	*0,68*	*0,68*	*0,68*	*0,25*	*0,93*
$\max \lvert \sigma_i \rvert$	ksi	24,99	24,94	24,98	25,00	24,04	25,00	25,00	*25,00*	*25,00*	*36,52*	*36,52*	*36,52*	25,00	*55,42*
w_{lim}	ksi	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25	0,25
σ_{lim}	ksi	25	25	25	25	25	25	25	25	25	25	25	25	25	25

Tabulka 6.9: Výsledky pro 72-prutovou konstrukci - spojitý problém

6.2 Hledání globálních optim

V předchozí podkapitole jsou zaznamenána optima nalezená v publikované literatuře. Některá optima jsou dobrými kandidáty na pozici globálního optima jako například u 25-prutové konstrukce. U některých konstrukcí je naopak rozdíl mezi spojitým a diskrétním optimem velký a tím pádem je zde vůle k nalezení lepšího optima diskrétního jako například u 72-prutové konstrukce.

počet prutů	10	25	52	72
počet uzlů	6	10	20	20
počet posunů	12	30	40	60
počet nenulových posunů	8	18	32	48
velikost matice tuhosti	8x8 (12x12)	18x18 (30x30)	32x32 (40x40)	48x48 (60x60)
počet skupin prutů	10	8	12	16
počet profilů v sadě	42	30	64	32
počet všech řešení metodou hrubé síly	42^{10} $= 1,7 \cdot 10^{16}$	30^{8} $= 6,6 \cdot 10^{11}$	64^{12} $= 4,7 \cdot 10^{21}$	32^{16} $= 1,2 \cdot 10^{24}$

Tabulka 6.10: Informace o úlohách a jejich velikosti

V tabulce 6.10 jsou znázorněny základní charakteristiky optimalizačních úloh. První část značí náročnost výpočtu omezujících podmínek, přičemž nejméně náročnou úlohou je 10-prutová konstrukce, srovnání viz tabulky 3.3 až 3.6. Druhá část znázorňuje počet možných řešení, pokud by se úloha řešila metodou hrubé síly. 25-prutová konstrukce je již rozdělena do skupin a zároveň sada s profily obsahuje nejméně prvků. Jedná se o kombinace s opakováním (uspořádané k-tice o n prvcích), počet všech řešení je tedy n^k, číselně viz tabulka. Jelikož se omezující podmínky v metodě větví a mezí počítají pouze mezi dolní a horní mezí účelové funkce, dá se předpokládat, že přestože je matice tuhosti větší než u 10-prutové konstrukce a výpočet omezujících podmínek na jednom řešení zabere více času, bude výpočet optimalizace 25-prutové konstrukce výpočetně nejméně náročný kvůli menšímu počtu potenciálních řešení.

6.2.1 Metoda hrubé síly

V metodě hrubé síly se počítají hodnoty omezujících podmínek pro všechna potenciální řešení. Pro pětiprutovou konstrukci je snadné ověřit správnost implementace algoritmu popsaného v podkapitole 4.2.1. Konstrukce je se svými 10^5 řešeními dostatečně malá a tím pádem je možné si všechna řešení uložit do jednoho pole včetně hodnot účelové funkce, tedy hmotností, a splnění omezujících podmínek. Nevyhovující řešení je pak možné vyřadit a vyhovující řešení seřadit dle jejich hmotnosti. Hodnota účelové funkce pro řešení s nejnižší hmotností je pak globálním optimem. Výsledek viz tabulka 6.11.

Pro 25-prutovou konstrukci, která má $6,6 \cdot 10^{11}$ potenciálních řešení metoda hrubé síly k nalezení globálního optima použít nelze kvůli potřebě neúměrně velkého výpočetního času. Pokud by byl použit nejrychlejší řešič soustavy lineárních rovnic s časem výpočtu 0,0118 sekund pro 1000 spuštění, trvalo by jen vyčíslení omezujících podmínek skoro 90 dní. K tomu je ještě nutné připočítat čas pro generování potenciálních řešení a obsluhu počítače. Tato metoda lze nicméně použít pro analýzu publikovaných optim a jejich okolí. V příloze D jsou uvedeny hypergrafy okolí nejlepších publikovaných řešení pro jednotlivé testovací konstrukce a v tabulkách 6.12 až 6.15 jsou uvedena nejlepší řešení nalezená touto metodou včetně řešení, ze kterých se vycházelo.

6.2.2 Metoda větví a mezí

5-prutová konstrukce

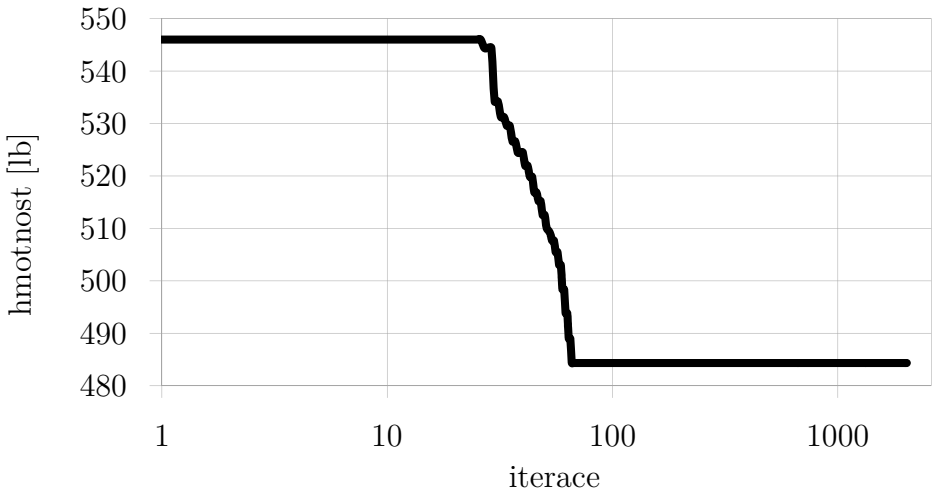
Metoda byla implementována v prostředí MATLAB a jazyce C/C++ v souladu s algoritmem popsaným v podkapitole 4.2.2 pro 5-prutovou konstrukci. Pro odhad dolní meze byla použita spojitá optimalizace procedurou `fmincon()` popsanou v podkapitole 4.3.1, výsledky viz tabulka 6.11. Pro odhad horní meze bylo zvoleno 0,23 lb, což je cca o 25 % více než je hodnota účelové funkce globálního optima získaného metodou hrubé síly. Srovnání optima získaného hrubou silou a metodou větví a mezí slouží k verifikaci algoritmu a jeho implementace. Metoda větví a mezí dokáže nalézt globální optimum.

Proměnná	Jednotky	diskrétní opt.		spojitá opt.
		hrubá síla	metoda větví a mezí	fmincon()
A_1	in^2	0,05	0,05	0,0500
A_2	in^2	0,01	0,01	0,01
A_3	in^2	0,06	0,06	0,0471
A_4	in^2	0,02	0,02	0,0167
A_5	in^2	0,01	0,01	0,01
m	lb	**0,179**	**0,179**	0,157
$\max \lvert w_j \rvert$	in	0,059	0,059	0,06
$\max \lvert \sigma_i \rvert$	ksi	59,371	59,371	60,006
w_{lim}	in	0,06	0,06	0,06
σ_{lim}	ksi	60	60	60

Tabulka 6.11: Výsledky pro pětiprutovou konstrukci

25-prutová konstrukce

Implementace metody větví a mezí byla provedena pro 25-prutovou konstrukci až na drobné úpravy stejně jako pro 5-prutovou konstrukci. Testována byla v prostředí MATLAB včetně paralelizace pomocí metody `spmd`. Pro dolní mez posloužila získaná



Obrázek 6.1: Snižování horní meze m_{max} pro 25-prutovou konstrukci

hodnota účelové funkce metodou `fmincon()` a pro horní mez nejhorší dostupná hodnota v literatuře [64] viz tabulka 6.6. Jednotlivé předem vygenerované kombinace se postupně posílaly na osm labů po padesáti. Vzniklo tedy $30^4/(8 \cdot 50) = 2025$ iterací. Předpokládaného optima bylo dosaženo při 66. iteraci. Úlohu však v tomto kroku nelze zastavit, neboť se hledá hodnota globálního optima, prostor se postupně musí prohledat celý. Též je nutné podotknout, že pokud by byly předem generované iterace načítány v obráceném pořadí, tedy odzadu dopředu (viz obrázek 5.3), bylo by optima dosaženo v iteraci jiné. Na obrázku 6.1 je znázorněn graf snižování horní meze m_{max}.

Proměnná	Jednotky	Pospíšilová B&B 2011 [62]	Kripka SA 2004 [43]	Pospíšilová Hrubá síla 2011 [62]	Pospíšilová `fmincon()` 2011 [62]	Perez PSO 2007 [60]
A_1	in^2	0,1	0,1	0,1	0,1	0,1
A_2	in^2	0,4	0,4	0,4	0,4214	0,4565
A_3	in^2	3,4	3,4	3,4	3,4	3,4
A_4	in^2	0,1	0,1	0,1	0,1	0,1
A_5	in^2	2,2	2,2	2,2	1,9166	1,9369
A_6	in^2	1,0	1,0	1,0	0,9656	0,9647
A_7	in^2	0,4	0,4	0,4	0,4706	0,4423
A_8	in^2	3,4	3,4	3,4	3,4	3,4
m	lb	**484,33**	**484,33**	**484,33**	483,82	483,84
$\max \lvert w_j \rvert$	in	0,35	0,35	0,35	6,13	6,15
$\max \lvert \sigma_i \rvert$	ksi	6,20	6,20	6,20	0,35	0,35
w_{lim}	in	0,35	0,35	0,35	40	40
σ_{lim}	ksi	40	40	40	0,35	0,35
Poznámka				4 kroky		

Tabulka 6.12: Výsledky pro 25-prutovou konstrukci včetně porovnání s publikovanými optimy

Tato hodnota určuje vždy dosud nejlepší nalezené řešení, proto tedy není možné, aby křivka grafu oscilovala. Graf lze tedy interpretovat jako konvergenci účelové funkce ke globálnímu optimu.

Získané řešení metodou větví a mezí se shoduje s řešením autora Kripky [43], který použil metodu simulovaného žíhání. Autor Kripka ale neprohledával celý prostor, proto si nemohl být jist, že nalezené optimum je globální a jestli ještě v prostoru neleží řešení lepší. Pokud jsou však srovnány výsledky v tabulce 6.12, kde lze nalézt jak řešení metodou větví a mezí, řešení Kripky, tak použité spojité řešení pro odhad dolní meze, dá se konstatovat, že hodnoty účelových funkcí řešení diskrétního a spojitého jsou velmi podobné a že nalezené řešení je potenciálně správné.

Výpočet trval na osmi procesech 15 dní, 6 hodin a 8 minut, proto je zřejmé, že MATLAB nebude vhodné používat pro ostatní konstrukce z důvodu dlouhého výpočetního času a z důvodu využití maximálně 12 procesů pomocí *Parallel Computing Toolbox*u.

Implementace proto byla provedena i v jazyce C/C++. Nejprve byl algoritmus testován jako sériový výpočet se stejnými hodnotami horní a dolní meze jako v MATLABu. Poté se spustil ještě s nejnižší možnou hodnotou dolní meze 33,072 liber (na všech prutech jsou profily 0,1 in^2) pro naprostou verifikaci globálního optima. Opět byl obdržen stejný výsledek jako je uveden v tabulce 6.12. Je tedy naprosto zřejmé, že prezentované optimum je skutečně globální. Dále byl výpočet vhodně upraven do paralelní verze a tyto kódy posloužily jako vzor pro konstrukce větší. Pro srovnání, výpočet s využitím jednoho procesu trval 22 hodin a 11 minut, při využití deseti procesů trval 5 hodin a 1 minutu.

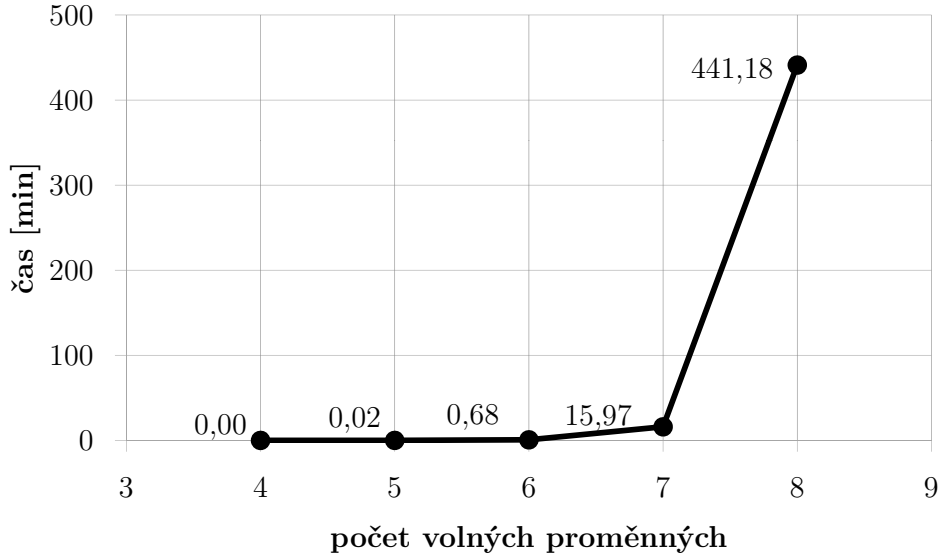
10-prutová konstrukce

Pro ostatní konstrukce byla metoda větví a mezí implementována v jazyce C/C++ sériově i paralelně s využitím MPI. Jelikož jsou úlohy větší něž 25-prutová konstrukce, byly zafixovány některé proměnné a ty se při každém nově spuštěném výpočtu postupně uvolňovaly. Cílem bylo alespoň odhadnout, jak velký výpočetní výkon bude zapotřebí mít k dispozici a jak dlouhou dobu výpočet potrvá. Zároveň bylo zaznamenáno nejlepší nalezené řešení, u kterého ovšem není zaručeno, že optimum je globální. K zafixování hodnot byly použity hodnoty proměnných nejlepšího publikovaného řešení.

Pro desetiprutovou konstrukci byla jako dolní mez použita hodnota získaná rutinou `fmincon` s hodnotou účelové funkce 5482,82 lb. Toto optimum sice vykazuje drobné přesahy hodnot omezujících podmínek, to ale u dolní meze nevadí. Podrobněji viz tabulka 6.13. Pro horní mez byla použita hodnota účelové funkce 5525,04 lb získané autorem Sousou a Takahashim [14]. K zafixování proměnných bylo použito řešení autorů Caie a Thieraufa [10] viz tabulka 6.13. Postupným uvolňováním se získalo řešení se dvěma zafixovanými proměnnými (v tabulce jsou zvýrazněny strojovým písmem) a osmi uvolněnými.

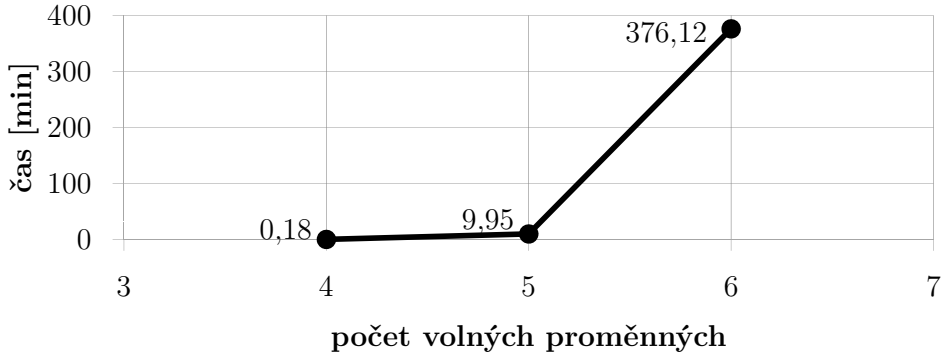
Proměnná	Jednotky	tato práce B&B	Cai ES 1996 [10]	tato práce Hrubá síla	tato práce fmincon()	Nosek DE 2008	Perez PSO 2007
A_1	in^2	33,5	33,5	33,5	32,235	30,473	33,50
A_2	in^2	1,62	1,62	1,62	1,620	0,100	0,10
A_3	in^2	22,9	22,9	22,9	23,296	23,173	22,76
A_4	in^2	14,2	14,2	14,2	15,262	15,210	14,42
A_5	in^2	1,62	1,62	1,62	1,620	0,100	0,10
A_6	in^2	1,62	1,62	1,62	1,620	0,561	0,10
A_7	in^2	7,97	7,97	7,97	8,307	7,464	7,53
A_8	in^2	22,9	22,9	22,9	22,687	21,089	20,46
A_9	in^2	22,0	22,0	22,0	21,584	21,527	20,40
A_{10}	in^2	1,62	1,62	1,62	1,620	0,100	0,10
m	lb	**5490,738**	**5490,738**	**5490,738**	5482,820	5060,925	5024,19
$\max \lvert w_j \rvert$	in	1,999	1,999	1,999	*2,000*	2,000	*2,039*
$\max \lvert \sigma_i \rvert$	ksi	14,197	14,197	14,197	13,677	24,998	*25,015*
w_{lim}	in	2	2	2	2	2	2
σ_{lim}	ksi	25	25	25	25	25	25
Poznámka		2 zafixované		2 kroky			

Tabulka 6.13: Výsledky pro 10-prutovou konstrukci včetně porovnání s publikovanými optimy



Obrázek 6.2: Nárůst rychlosti pro postupné uvolnění proměnných 10-prutové konstrukce (k výpočtu bylo použito 10 procesů)

Jelikož je hodnota účelové funkce spojitého řešení blízká řešení diskrétnímu, je pravděpodobné, že se hodnota globálního optima již moc neodchýlí případně zůstane stejná jako se zafixovanými proměnnými. Toto řešení je totiž získáno více autory nezávisle na sobě viz tabulka 6.1. Za zmínku stojí ještě přenastavení dolní meze na



Obrázek 6.3: Nárůst rychlosti pro postupné uvolnění proměnných 52-prutové konstrukce (k výpočtu bylo použito 10 procesů)

hodnotu účelové funkce autorů Pereze a Behdinana. Při zafixování tří proměnných bylo získáno stejné řešení jako s původní dolní mezí, čas výpočtu se však zvýšil z necelých 16 minut na 3,4 hodiny! Zde je názorně vidět, že při znalosti kvalitního odhadu dolní meze je možné enormně ušetřit výpočetní čas. Na grafu z obrázku 6.2 je opět znázorněn nárůst rychlosti pro postupné uvolnění proměnných.

52-prutová konstrukce

Padesátidvouprutová konstrukce má jako jediná pouze jednu omezující podmínku, a to ve formě napětí. Při použití metody `fmincon()` s několika náhodnými startovacími body řešení vykazují velký rozptyl. Jelikož není dostupné žádné publikované řešení pro spojitou optimalizaci, aby se mohla získaná řešení porovnat, přistoupilo se k použití `fmincon()` se startovacím bodem z optima autora Gigera, což je nejlepší optimum publikované v dostupné literatuře. Výsledkem je řešení s hodnotou účelové funkce 1826,6429 kg, které sice vykazuje drobný přesah omezující podmínky, ale jelikož se ale jedná o dolní mez účelové funkce, nevyhovující řešení nevadí. Přestože je tato hodnota poměrně dobrá, není tato konstrukce dostatečně prozkoumaná. Proto se přistoupilo ke snížení hodnoty dolní meze na 1500 kg. Dá se předpokládat, že tato hodnota již bude dostatečná pro nalezení globálního optima. Jako horní mez se vzala nejlepší nalezená hodnota v literatuře s hmotností 1903,366 kg v [28], neboť má napětí menší hodnotu než u řešení autorů Lemongeho a Barbosy. Toto řešení se zároveň použilo i pro zafixování proměnných.

Úloha se spustila několikrát s postupným uvolněním zafixovaných proměnných. Při zafixování osmi proměnných bylo nalezeno lepší optimum než je publikované v literatuře a zůstalo stejné, i když se uvolnily další dvě proměnné. K dalšímu uvolnění proměnných by ale byl potřeba větší výpočetní výkon respektive delší výpočetní čas, proto k němu prozatím v této práci nebylo přistoupeno. Dá se nicméně předpokládat, že zde bude prostor pro ještě lepší optimum, neboť hodnota účelové funkce spojité

Proměnná	Jednotky	tato práce B&B	Giger 2006 [28]	tato práce Hrubá síla	tato práce fmincon()
A_1	in^2	`4658,055`	4658,055	4658,055	4406,1956
A_2	in^2	`1161,288`	1161,288	1161,288	1122,7585
A_3	in^2	`494,193`	494,193	494,193	280,8333
A_4	in^2	`3303,219`	3303,219	3303,219	3359,5486
A_5	in^2	`940,000`	940,000	940,000	890,1741
A_6	in^2	`494,193`	494,193	494,193	361,7214
A_7	in^2	2238,705	2238,705	2238,705	2279,5500
A_8	in^2	1008,385	1008,385	1008,385	976,4743
A_9	in^2	494,193	363,225	363,225	345,4084
A_{10}	in^2	1283,868	1283,868	1283,868	1308,5520
A_{11}	in^2	1161,288	1161,288	1161,288	1063,1153
A_{12}	in^2	494,193	641,289	641,289	425,6548
m	kg	**1902,606**	1903,3664	1903,3664	1826,6429
$\max \lvert\sigma_i\rvert$	MPa	179,765	179,783	179,783	*180,001*
σ_{lim}	MPa	180	180	180	180
Poznámka		6 zafixovaných		1 krok	

Tabulka 6.14: Výsledky pro 52-prutovou konstrukci včetně porovnání s publikovaným optimem

optimalizace, přestože se nejedná o optimum z důvodu nesplnění omezující podmínky, je výrazně nižší. Řešení viz tabulka 6.14. Na obrázku 6.3 je znázorněn nárůst rychlosti pro postupné uvolnění proměnných. Dá se předpokládat, že výpočetní čas bude při uvolnění všech proměnných enormní. Proto bude nejspíše nutné zvýšit hodnotu dolní meze za dalšího zkoumání okolí dosud nejlepšího nalezeného řešení.

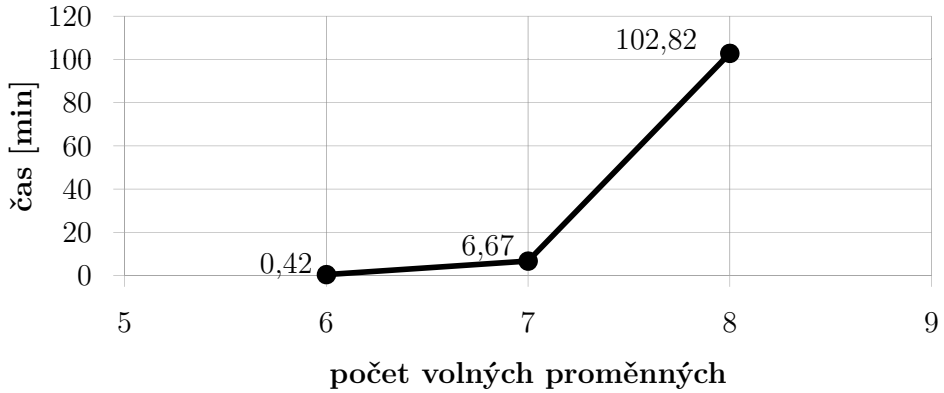
72-prutová konstrukce

Dolní mez účelové funkce byla u metody větví a mezí nastavena na 379,61 lb, což je nejlepší získaná hodnota rutinou `fmincon()`. Jednotlivé hodnoty se mezi sebou téměř neliší a zároveň je toto řešení blízké řešení autora Fleuryho. Pro horní mez bylo převzato řešení autorů Wu a Chowa s hodnotou 400,66 lb. Toto řešení zároveň posloužilo i k zafixování jednotlivých proměnných.

Úloha se opět řešila s postupným uvolňováním zafixovaných proměnných. Opět bylo nalezeno lepší řešení než je publikované, přesto nebylo tak dobré jako řešení hrubou silou na okolí publikovaného optima při snížení respektive zvýšení hodnot proměnných maximálně o jeden profil v sadě. Výsledky viz tabulka 6.15. Na grafu z obrázku 6.4 je opět znázorněn nárůst rychlosti pro postupné uvolnění proměnných. Tato úloha již nemá tak rychlou tendenci s přibývajícími uvolněnými proměnnými, ale problém je ve větším počtu potenciálních řešení.

Proměnná	Jednotky	tato práce B&B	Wu 1995 GA [88]	tato práce Hrubá síla	tato práce fmincon()	Fleury 1980 [21]
A_1	in^2	0,2	0,2	0,2	0,156	0,157
A_2	in^2	0,5	0,5	0,5	0,546	0,536
A_3	in^2	0,5	0,5	0,4	0,410	0,410
A_4	in^2	0,7	0,7	0,6	0,570	0,568
A_5	in^2	0,5	0,5	0,6	0,524	0,507
A_6	in^2	0,5	0,5	0,6	0,517	0,520
A_7	in^2	0,1	0,1	0,1	0,1	0,1
A_8	in^2	0,2	0,2	0,1	0,1	0,1
A_9	in^2	1,2	1,3	1,3	1,268	1,28
A_{10}	in^2	0,5	0,5	0,5	0,512	0,515
A_{11}	in^2	0,1	0,2	0,1	0,1	0,1
A_{12}	in^2	0,1	0,1	0,1	0,1	0,1
A_{13}	in^2	1,8	1,5	1,6	1,886	1,897
A_{14}	in^2	0,6	0,7	0,6	0,512	0,516
A_{15}	in^2	0,1	0,1	0,1	0,1	0,1
A_{16}	in^2	0,1	0,1	0,1	0,1	0,1
m	lb	389,93	400,66	**389,08**	379,61	379,67
$\max \lvert w_j \rvert$	in	0,25	0,25	0,25	*0,25*	0,25
$\max \lvert \sigma_i \rvert$	ksi	20,35	20,38	20,71	25,00	24,99
w_{lim}	in	0,25	0,25	0,25	0,25	0,25
σ_{lim}	ksi	25	25	25	25	25
Poznámka		8 zafixovaných		1 krok		

Tabulka 6.15: Výsledky pro 72-prutovou konstrukci včetně porovnání s publikovanými optimy



Obrázek 6.4: Nárůst rychlosti pro postupné uvolnění proměnných 72-prutové konstrukce (k výpočtu bylo použito 10 procesů)

Kapitola 7

Závěr

Hledání globálních optim není jednoduché z hlediska nutnosti použití velkého výpočetního výkonu případně enormně velkého výpočetního času. K hledání na menších konstrukcích, jako je pětiprutová konstrukce, dobře poslouží metoda hrubé síly, která je snadná na implementaci a je možné s ní vypočítat všechna zadání konstrukce. Ty se mohou během výpočtu postupně ukládat a lze tak získat širší ponětí o optimalizační úloze (například jestli má úloha více globálních optim, jak jsou rozloženy skladby pro vyhovující okrajové podmínky a podobně).

Naproti tomu konstrukce, které se pro rozměrovou optimalizaci běžně používají, již metodou hrubé síly optimalizovat v přijatelném čase nelze. Pro ně lze použít lepší optimalizační metodu, a to metodu větví a mezí. Tato metoda používá k vyčlenění menšího prostoru, kde může ležet globální optimum, dolní a horní mez účelové funkce a v tomto prostoru se pak řešení efektivně vyhledává. Dolní mez je možné určit pomocí některé metody vhodné pro spojitou optimalizaci. Tyto metody jsou již po dlouhá léta důkladně zkoumány. Přestože je potřeba mít k metodě větví a mezí dolní mez co nejblíže spojitému globálnímu optimu, stačí se k této hodnotě zespodu přiblížit a mít tak nějakou relativně dobrou dolní mez, při které nehrozí ztráta globálního optima, ale zároveň bude úloha spočitatelná v relativně přijatelném čase. Horní mez je možné spočítat například heuristickou metodou, která je rychlá a snadno vyhledá lokální optimum. Tato mez se v rámci výpočtu snižuje, čímž se opět zmenší prohledávaný prostor mezi mezemi. Čím užší jsou pak na začátku tyto meze, tím je metoda větví a mezí efektivnější, jelikož se problém nerozvětví do tolika podproblémů.

Pro 5-prutovou konstrukci se podařilo najít globální optimum jak metodou hrubé síly, tak metodou větví a mezí, a tato shoda může být považována za kontrolu správnosti algoritmu metody větví a mezí a její implementace. Pro 25-prutovou konstrukci bylo též možné v relativně krátkém čase získat globální optimum metodou větví a mezí jak v prostředí MATLAB při paralelním výpočtu, tak v jazyce C/C++ při sériovém i paralelním výpočtu s využitím protokolu MPI. Toto globální optimum bylo získáno s dolní mezí spočítanou nelineárním programováním implementovaným v rámci MATLABu

a horní mezí nastavenou dle publikovaného lokálního optima v literatuře. Pro verifikaci tohoto globálního optima byla též nastavena dolní mez na svou nejnižší možnou hodnotu. Tím sice enormně vzrostl výpočetní čas (49 dní, 10 hodin a 53 minut) oproti původně nastavené dolní mezi (22 hodin a 11 minut), ale úloha ještě byla spočitatelná v reálném čase. Nárůst času je dán tím, že se muselo vyhodnotit více omezujících podmínek, které se při lepším odhadu dolní meze nevyhodnocují.

Pro další konstrukce bylo přistoupeno k zafixování některých proměnných k otestování chování konstrukce a odhadu délky výpočetního času. Při analýze chování bylo pro 52-prutovou a 72-prutovou konstrukci nalezeno lepší řešení než bylo nalezeno v námi dostupné publikované literatuře. Nicméně se předpokládá, že po uvolnění všech proměnných budou hodnoty účelové funkce ještě lepší z důvodu velkého rozdílu mezi hodnotou účelové funkce spojité a diskrétní úlohy. K tomu však bude potřebný ještě větší počítačový výkon než byl pro tuto práci k dispozici.

Dalším nezbytným aspektem pro výpočet globálních optim je právě velký výpočetní výkon. Bez možnosti paralelního výpočtu globální optima v blízké době rozhodně nebude možné získávat, neboť již nevzrůstá výkon v rámci jednojádrového procesoru, ale procesory se vyrábí s více jádry případně je počítač vybaven více takovýmito procesory. Začíná se též počítat na grafických kartách, které zvládnou jednoduché operace, například program CUDA vyvíjený společností NVIDIA. Další kapitolou je paralelní počítání v rámci clusteru případně jiného zapojení počítačů do sítě jako jsou gridy nebo cloudy. Úloha se pak dá při vhodném naprogramování a porozumění problému dobře rozdělit a je možné získat skoro lineární nárůst rychlosti.

Pro výpočet v rámci této práce byl k dispozici pouze malý výpočetní výkon, přesto bylo získáno globální optimum pro 25-prutovou konstrukci. Jeho získání je dobrým předpokladem pro to, že se s větším výkonem (např. 10 počítačů vybavených osmi jádry - tedy 80 procesů) a dobrým nastavením dolní a horní meze podaří najít globální optima i pro konstrukce ostatní. Na dvou konstrukcích byla však získána v rámci tohoto výzkumu lokální optima lepší než jsou nalezena v publikované literatuře. Získaná globální optima všech konstrukcí po dokončení výpočtů poslouží jako etalon kvality pro zlepšování nových ale i stávajících optimalizačních algoritmů.

Literatura

[1] H. Adeli and O. Kamal. Efficient optimization of space trusses. *Computers & Structures*, 24(3):501–511, 1986.

[2] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, třetí edition, 1999.

[3] J. S. Arora. Methods for discrete variable structural optimization. In S. A. Burns, editor, *Recent Advantages in Optimal Structural Design*, chapter 1, pages 1–40. American Society of Civil Engineers, 2002.

[4] E. Barbieri and M. Lomhardi. Minimum weight shape and size optimization of truss structures made of uncertain materials. *Structural and Multidisciplinary Optimization*, 16:147–154, 1998.

[5] H. J. C. Barbosa and A. C. C. Lemonge. A new adaptive penalty scheme for genetic algorithms. *Information Sciences*, 156:215–251, November 2003.

[6] H. J. C. Barbosa, A. C. C. Lemonge, and C. C. H. Borges. A genetic algorithm encoding for cardinality constraints and automatic variable linking in structural optimization. *Engineering Structures*, 30(12):3708–3723, 2008.

[7] M. P. Bendsøe. *Optimization of structural topology, shape and material*. Springer-Verlag, 1 edition, 1995.

[8] Z. Bittnar. *Metody numerické analýzy konstrukcí*. Skriptum. České vysoké učení technické v Praze, 1983.

[9] F. Bubeník, M. Pultar, and I. Pultarová. *Matematické vzorce a metody*. Skriptum. České vysoké učení technické v Praze, 1997.

[10] J. Cai and G. Thierauf. Evolution strategies for solving discrete optimization problems. *Advances in Engineering Software*, 25(2-3):177–183, 1996.

[11] J. Cai and G. Thierauf. A parallel evolution strategy for solving discrete structural optimization. *Advances in Engineering Software*, 27(1-2):91 – 96, 1996.

[12] C. Camp, S. Pezeshk, and G. Cao. Optimized design of two-dimensional structures using a genetic algorithm. *Journal of Structural Engineering*, 124(5):551–559, 1998.

[13] C. A. C. Coello. Discrete optimization of trusses using genetic algorithms. In F. G. A. J. G. Chen and D. L. Crabtree, editors, *EXPERSYS-94: Expert Systems Applications and Artificial Intelligence*, Technology Transfer Series, pages 331–336. I.I.T.T. International, 1994.

[14] F. L. de Sousa and W. K. Takahashi. Discrete optimal design of trusses by generalized extremal optimization. In *6th World Congresses of Structural and Multidisciplinary Optimization*, Rio de Janeiro, Brazil, May - June 2005.

[15] J. Dongarra, R. Pozo, and D. Walker. LAPACK++ V. 1.1: High performance linear algebra users' guide.
`http://math.nist.gov/lapack++/lapackppman1_1.ps.gz`, 1996.

[16] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. *Metaheuristics for Hard Optimization: Methods and Case Studies*. Springer, 1st edition, December 2005.

[17] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer, 1st edition, 2003.

[18] F. Erbatur, O. Hasançebi, I. Tütüncü, and H. Kılıç. Optimal design of planar and space structures with genetic algorithms. *Computers & Structures*, 75(2):209–224, 2000.

[19] J. Fish and T. Belytschko. *A First Course in Finite Elements*. John Wiley & Sons, Ltd, April 2007.

[20] C. Fleury. A unified approach to structural weight minimization. *Computer Methods in Applied Mechanics and Engineering*, 20(1):17–38, 1979.

[21] C. Fleury and L. A. Schmit. Dual methods and approximation concepts in structural synthesis. Technical Report NASA-CR-3226, National Aeronautics and Space Administration, Scientific and Technical Information Branch, 1980.

[22] M. P. I. Forum. Mpi: A message-passing interface standard, version 2.2. Technical report, University of Tennessee, Knoxville, Tennessee, September 2009.

[23] R. L. Fox and L. A. Schmit. Advances in the integrated approach to structural synthesis. *Journal of Spacecraft and Rockets*, 3(6):858–866, June 1966.

[24] M. Galante. Genetic algorithms as an approach to optimize real-world trusses. *International Journal for Numerical Methods in Engineering*, 39(3):361–382, February 1996.

[25] S. Ganzerli and C. P. Pantelides. Optimum structural design via convex model superposition. *Computers & Structures*, 74(6):639 – 647, 2000.

[26] R. A. Gellatly and P. V. Marcal. Investigation of advanced spacecraft structural design technology. NASA-CR 89637, NASA, 1967.

[27] M. R. Ghasemi, E. Hinton, and R. Wood. Optimization of trusses using genetic algorithms for discrete and continuous variables. *Engineering computations*, 16(3):272–301, 1999.

[28] M. Giger and P. Ermanni. Evolutionary truss topology optimization using a graph-based parameterization concept. *Structural and Multidisciplinary Optimization*, 32:313–326, 2006.

[29] A. A. Groenwold, N. Stander, and J. A. Snyman. A pseudo-discrete rounding method for structural optimization. *Structural and Multidisciplinary Optimization*, 11:218–227, 1996.

[30] W. Guang-Yuan, Z. Zheng-Yuan, and H. Da. A two-phase optimization method for minimum-weight design of trusses. *Engineering Optimization*, 8(1):55–67, 1984.

[31] W. Gutkowski, J. Bauer, and Z. Iwanow. Discrete structural optimization. *Computer Methods in Applied Mechanics and Engineering*, 51(1-3):71–78, 1985.

[32] A. Hadidi, S. K. Azad, and S. K. Azad. Structural optimization using artificial bee colony algorithm. In *2nd International Conference on Engineering Optimization*, Lisbon, Portugal, September 2010.

[33] R. T. Haftka. Simultaneous analysis and design. *AIAA Journal*, 23(7):1099–1103, July 1985.

[34] O. Hasançebi, S. Çarbaş, E. Doğan, F. Erdal, and M. P. Saka. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Computers & Structures*, 87(5-6):284–302, 2009.

[35] O. Hasançebi and F. Erbatur. Evaluation of crossover techniques in genetic algorithm based optimum structural design. *Computers & Structures*, 78(1-3):435 – 448, 2000.

[36] O. Jiroušek. Metoda konečných prvků: poznámky k přednáškám. `http://mech.fd.cvut.cz/education/master/k618y2m1/download/ymkp_fem.pdf`, 2006.

[37] G. Jivotovski. A gradient based heuristic algorithm and its application to discrete optimization of bar structures. *Structural and Multidisciplinary Optimization*, 19:237–248, 2000.

[38] B. Kang, W. Choi, and G. Park. Structural optimization under equivalent static loads transformed from dynamic loads based on displacement. *Computers & Structures*, 79(2):145 – 154, 2001.

[39] A. Kaveh and M. Shahrouzi. Dynamic selective pressure using hybrid evolutionary and ant system strategies for structural optimization. *International Journal for Numerical Methods in Engineering*, 73(4):544–563, 2008.

[40] A. Kaveh and S. Talatahari. A particle swarm ant colony optimization for truss structures with discrete variables. *Journal of Constructional Steel Research*, 65(8-9):1558–1568, August 2009.

[41] A. Kaveh and S. Talatahari. Particle swarm optimizer, ant colony strategy and harmony search scheme hybridized for optimization of truss structures. *Computers & Structures*, 87(5-6):267–283, 2009.

[42] U. Kirsch. Layout optimization using reduction and expansion processes. *First World Congress of Structural and Multidisciplinary Optimization*, 1995.

[43] M. Kripka. Discrete optimization of trusses by simulated annealing. *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, 26(2):170–173, April-June 2004.

[44] L. Lamberti. An efficient simulated annealing algorithm for design optimization of truss structures. *Computers & Structures*, 86(19-20):1936–1953, 2008.

[45] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, July 1960.

[46] J. Lee and P. Hajela. Application of classifier systems in improving response surface based approximations for design optimization. *Computers & Structures*, 79(3):333 – 344, 2001.

[47] K. S. Lee. Standard harmony search algorithm for structural design optimization. In Z. Geem, editor, *Harmony Search Algorithms for Structural Design Optimization*, volume 239 of *Studies in Computational Intelligence*, pages 1–49. Springer Berlin / Heidelberg, 2009.

[48] K. S. Lee and Z. W. Geem. A new structural optimization method based on the harmony search algorithm. *Computers & Structures*, 82(9-10):781–798, 2004.

[49] K. S. Lee, Z. W. Geem, S. ho Lee, and K. woong Bae. The harmony search heuristic algorithm for discrete structural optimization. *Engineering Optimization*, 37(7):663–684, 2005.

[50] A. C. C. Lemonge and H. J. C. Barbosa. An adaptive penalty scheme for genetic algorithms in structural optimization. *International Journal for Numerical Methods in Engineering*, 59(5):703–736, 2004.

[51] R. Levy, U. Kirsch, and S. Liu. Reanalysis of trusses using modified initial designs. *Structural and Multidisciplinary Optimization*, 19:105–112, 2000.

[52] L. Li, Z. Huang, F. Liu, and Q. Wu. A heuristic particle swarm optimizer for optimization of pin connected structures. *Computers & Structures*, 85(7-8):340–349, 2007.

[53] L. J. Li, Z. B. Huang, and F. Liu. A heuristic particle swarm optimization method for truss structures with discrete variables. *Computers & Structures*, 87(7-8):435–443, 2009.

[54] M. Lombardi. Optimization of uncertain structures using non-probabilistic models. *Computers & Structures*, 67(1-3):99 – 103, 1998.

[55] P. Luszczek. Parallel programming in MATLAB. `http://web.eecs.utk.edu/~luszczek/pubs/parallelmatlab.pdf`, July 2009.

[56] Message Passing Interface Forum. Message passing interface forum. `http://www.mpi-forum.org/`, 2011.

[57] D. Ming-Zhu. An improved templeman's algorithm for the optimum design of trusses with discrete member sizes. *Engineering Optimization*, 9(4):303–312, 1986.

[58] J. Nosek. Stochastická optimalizace návrhu konstrukcí. Master's thesis, České vysoké učení technické v Praze, 2008.

[59] C. Pantelides and B. C. Booth. Computer-aided design of optimal structures with uncertainty. *Computers & Structures*, 74(3):293 – 307, 2000.

[60] R. E. Perez and K. Behdinan. *Swarm Intelligence, Focus on Ant and Particle Swarm Optimization*, chapter Particle Swarm Optimization in Structural Design, pages 373–394. Itech Education and Publishing, Vienna, Austria, December 2007. ISBN: 978-3-902613-09-7.

[61] A. Pospíšilová. Analýza tradičních příkladů rozměrové optimalizace. Bakalářská práce, České vysoké učení technické v Praze, 2010.

[62] A. Pospíšilová and M. Lepš. Globální optima testovacích konstrukcí rozměrové optimalizace. Odesláno k publikaci ve Stavebním obzoru, 2011.

[63] M. Pyrz and J. Zawidzka. Optimal discrete truss design using improved sequential and genetic algorithm. *Engineering Computations*, 18(8):1078 – 1090, 2001.

[64] S. Rajeev and C. S. Krishnamoorthy. Discrete optimization of structures using genetic algorithms. *Journal of Structural Engineering*, 118(5):1233–1250, 1992.

[65] X. Renwei and L. Peng. Structural optimization based on second-order approximations of functions and dual theory. *Computer Methods in Applied Mechanics and Engineering*, 65(2):101 – 114, 1987.

[66] K. C. Sarma and H. Adeli. Fuzzy genetic algorithm for optimization of steel structures. *Journal of Structural Engineering*, 126(5):596–604, 2000.

[67] L. A. Schmit and H. Miura. Approximation concepts for efficient structural synthesis. Technical Report NASA-CR-2552, National Aeronautics and Space Administration, 1976.

[68] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. Technical report, School od Computer Science, Carnegie Mellon University, 1994.

[69] C. J. Shih. Fuzzy and improved penalty approaches for multiobjective mixed-discrete optimization in structural systems. *Computers & Structures*, 63(3):559–565, 1997.

[70] SIFEL group. Homepage of SIFEL. `http://mech.fsv.cvut.cz/~sifel/`, 2010.

[71] O. Sigmund and M. P. Bendsœ. *Topology Optimization: Theory, Methods and Applications*. Springer-Verlag, 2nd edition, 2003. ISBN 978-3-540-42992-0.

[72] G. Steven. Product and system optimization in engineering simulation. *FENet Newsletter*, 2003.

[73] P. Sváček and M. Feistauer. *Metoda konečných prvků*. Skriptum. České vysoké učení technické v Praze, 2006.

[74] K. Terai. Application of optimality criteria in structural synthesis. Technical Report NASA-CR-138612, National Aeronautics and Space Administration, 1974.

[75] The MathWorks. Constrained nonlinear optimization algorithms: Optimization algorithms and examples (Optimization Toolbox™). `http://www.mathworks.com/help/toolbox/optim/ug/brnoxzl.html`, 2011.

[76] The MathWorks. Find minimum of constrained nonlinear multivariable function - matlab. `http://www.mathworks.com/help/toolbox/optim/ug/fmincon.html`, 2011.

[77] The MathWorks. Matlab - the language of technical computing. `http://www.mathworks.com/products/matlab/`, 2011.

[78] The MathWorks. Matlab distributed computing server. `http://www.mathworks.com/products/distriben/index.html`, 2011.

[79] The MathWorks. Matlab R2011b: Mathematics. `http://www.mathworks.com/access/helpdesk/help/pdf_doc/matlab/math.pdf`, 2011.

[80] The MathWorks. Parallel computing toolbox - matlab. `http://www.mathworks.com/products/parallel-computing/`, 2011.

[81] The MathWorks. Product support: MEX-files guide. `http://www.mathworks.com/support/tech-notes/1600/1605.html`, December 2011.

[82] W. H. Tong and G. R. Liu. An optimization procedure for truss structures with discrete design variables and dynamic constraints. *Computers & Structures*, 79(2):155–162, 2001.

[83] V. B. Venkayya. Design of optimum structures. *Computers & Structures*, 1(1-2):265–309, 1971.

[84] V. B. Venkayya, N. S. Khot, and V. S. Reddy. Optimization of structures based on the study of energy distribution. Technical Report AFFDL-TR-68-150, Air Force Flight Dynamics Laboratory, Wright Patterson AFB,OH,45433, 1968.

[85] R. Vondráček. DSSinC - direct sparse solver in c++. `http://www.femcad.com/DSSinC/`, 2008.

[86] R. Vondráček. *Use of a Sparse Direct Solver in Engineering Applications of the Finite Element Method*. PhD thesis, České vysoké učení technické v Praze, 2008.

[87] S.-J. Wu and P.-T. Chow. Integrated discrete and configuration optimization of trusses using genetic algorithms. *Computers & Structures*, 55(4):695–702, 1995.

[88] S.-J. Wu and P.-T. Chow. Steady-state genetic algorithms for discrete optimization of trusses. *Computers & Structures*, 56(6):979–991, 1995.

[89] W. Xicheng and M. Guixu. A parallel iterative algorithm for structural optimization. *Computer Methods in Applied Mechanics and Engineering*, 96(1):25–32, 1992.

[90] X. Yu, S. Zhang, and E. Johnson. A discrete post-processing method for structural optimization. *Engineering with Computers*, 19:213–220, 2003.

[91] M. Zhou and G. I. N. Rozvany. DCOC: An optimality criteria method for large systems part II: Algorithm. *Structural and Multidisciplinary Optimization*, 6:250–262, 1993.

Příloha A

Přehled užitých anglosaských jednotek s převodem do SI soustavy

Jednotka	Název anglický	Název český	Převod do SI soustavy
in	inch	palec	1 in = 25,4 mm
psi	pound per square inch	libra síly na čtverečný palec	1 psi = 6.894,757 Pa
kp	kip	-	1 kip = 4.448,222 N
lb/in^3	pound per cubic inch	libra síly na kubický palec	1 lb/in^3 = 27679,905 kg/m^3

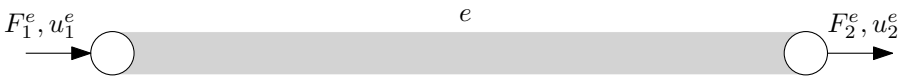
Tabulka A.1: Přehled užitých anglosaských jednotek s převodem do SI soustavy

Příloha B

Odvození deformační varianty MKP

Deformační varianta metody konečných prvků pro tažený respektive tlačený prut byla již mnohokrát odvozena a prezentována. Pro doplnění zde však budou uvedeny základní myšlenky a vzorce.

B.1 Lokální matice tuhosti pro 1D prut



Obrázek B.1: Prut orientovaný v ose x

Je-li prut zaveden jako na obrázku B.1 a uvažuje-li se tah jako kladný, pak lze uzlové síly vyjádřit jako $F_1^e = -\sigma_x \cdot A$ a $F_2^e = \sigma_x \cdot A$, kde σ_x je napětí a A je plocha příčného řezu. Dosazením do Hookova zákona $\sigma_x = E \cdot \varepsilon$, kde E je modul pružnosti, a vyjádřením poměrného prodloužení jako $\varepsilon_x = \frac{\Delta l}{l}$, kde l je délka prutu, a proudloužení jako $\Delta l = u_{2x} - u_{1x}$, kde u_{1x} a u_{2x} jsou koncové posuny uzlů, lze získat vztahy

$$F_1^e = -E \cdot \frac{u_2^e - u_1^e}{l} \cdot A, \tag{B.1}$$

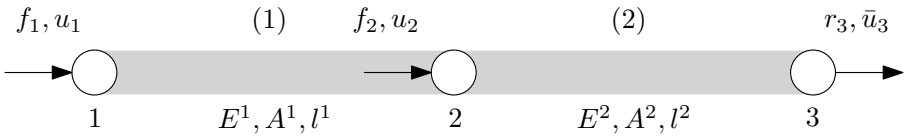
$$F_2^e = E \cdot \frac{u_2^e - u_1^e}{l} \cdot A. \tag{B.2}$$

Maticově pak

$$\left\{ \begin{array}{c} F_1^e \\ F_2^e \end{array} \right\} = \left[\begin{array}{cc} k^e & -k^e \\ -k^e & k^e \end{array} \right] \left\{ \begin{array}{c} u_1^e \\ u_2^e \end{array} \right\}, \text{kde } k^e = \frac{E \cdot A}{l}, \tag{B.3}$$

a zkráceně $f^e = K^e \cdot r^e$,
kde k^e má význam tuhosti prutu, K^e je lokální matice tuhosti, f^e je vektor uzlových zatížení prutu a r^e je vektor posunů.

B.2 Globální matice tuhosti pro 1D konstrukci



Obrázek B.2: Model dvouprutové příhradové konstrukce

Globální matice tuhosti nepopisuje pouze jeden prut jako matice lokální, ale celou konstrukci. Získá se lokalizací pomocí lokalizační matice viz [61] anebo umístěním lokální matice do globální pomocí lokálních a globálních kódových čísel. Je-li zavedena dvouprutová konstrukce tak, jako na obrázku B.2, pak lokální matice tuhosti mají následující podobu

$$K^1 = \begin{matrix} \\ 1 \\ 2 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 \end{matrix} \\ \begin{pmatrix} k^1 & -k^1 \\ -k^1 & k^1 \end{pmatrix} \end{matrix}, \; K^2 = \begin{matrix} \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 2 & 3 \end{matrix} \\ \begin{pmatrix} k^2 & -k^2 \\ -k^2 & k^2 \end{pmatrix} \end{matrix}, \tag{B.4}$$

a globální matice

$$K = \begin{matrix} \\ 1 \\ 2 \\ 3 \end{matrix} \begin{matrix} \begin{matrix} 1 & 2 & 3 \end{matrix} \\ \begin{pmatrix} k^1 & -k^1 & 0 \\ -k^1 & k^1 + k^2 & -k^2 \\ 0 & -k^2 & k^2 \end{pmatrix} \end{matrix}. \tag{B.5}$$

B.3 Regularizace matice tuhosti

Matice tuhosti, např. tak, jak je zavedená v rovnici B.5, je sama o sobě singulární. Pokud je však předepsán dostatečný počet okrajových podmínek, tedy například dostatečné a vhodné podepření konstrukce, dochází k regularizaci celé soustavy

$$\begin{bmatrix} K_{uu} & K_{up} \\ K_{pu} & K_{pp} \end{bmatrix} \begin{Bmatrix} u \\ \bar{u} \end{Bmatrix} = \begin{Bmatrix} f \\ r \end{Bmatrix}, \tag{B.6}$$

kde u jsou neznámé posuny, $\bar{u}$ jsou předepsané posuny, f jsou uzlová zatížení a r jsou reakce v podporách. Úpravami pak lze dojít k předpisu pro výpočet neznámých posunů u a reakcí r

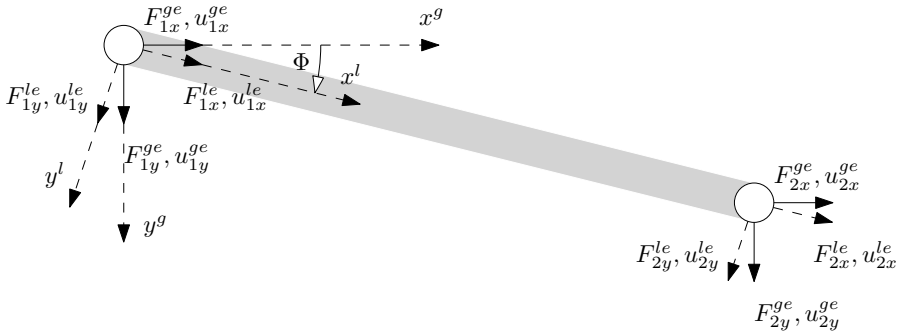
$$u = K_{uu}^{-1}(f - K_{up} \cdot \bar{u}). \tag{B.7}$$

$$r = K_{pu} \cdot K_{uu}^{-1}(f - K_{up} \cdot \bar{u}) + K_{pp} \cdot \bar{u}. \tag{B.8}$$

Pro konstrukci na obrázku B.2 pak soustava vypadá takto:

$$\left[\begin{array}{cc|c} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ \hline k_{31} & k_{32} & k_{33} \end{array}\right] \left\{\begin{array}{c} u_1 \\ u_2 \\ \hline \bar{u}_3 \end{array}\right\} = \left\{\begin{array}{c} f_1 \\ f_2 \\ \hline r_3 \end{array}\right\}. \tag{B.9}$$

B.4 Matice tuhosti pro 2D prut



Obrázek B.3: Prut natočený o úhel Φ

Je-li zaveden prut ve 2D jako na obrázku B.3, jsou neznámými posun ve vodorovném směru a svislém směru. Lokální matice tuhosti ve 2D vznikne rozšířením lokální matice pro 1D (B.3)

$$\left\{\begin{array}{c} F_{1x}^{le} \\ F_{1y}^{le} \\ F_{2x}^{le} \\ F_{2y}^{le} \end{array}\right\} = \left[\begin{array}{cccc} k^e & 0 & -k^e & 0 \\ 0 & 0 & 0 & 0 \\ -k^e & 0 & k^e & 0 \\ 0 & 0 & 0 & 0 \end{array}\right] \left\{\begin{array}{c} u_{1x}^{le} \\ u_{1y}^{le} \\ u_{2x}^{le} \\ u_{2y}^{le} \end{array}\right\}. \tag{B.10}$$

Přenásobením transformační maticí viz (B.11) lokální matice tuhosti

$$T^T = \left[\begin{array}{cccc} \cos\Phi^e & \sin\Phi^e & 0 & 0 \\ -\sin\Phi^e & \cos\Phi^e & 0 & 0 \\ 0 & 0 & \cos\Phi^e & \sin\Phi^e \\ 0 & 0 & -\sin\Phi^e & \cos\Phi^e \end{array}\right] \tag{B.11}$$

$T^T K^e T$ je získána globální matice tuhosti K^{ge}

$$K^{ge} = \frac{EA}{l} \left[\begin{array}{cccc} \cos^2\Phi^e & \cos\Phi^e \sin\Phi^e & -\cos^2\Phi^e & -\cos\Phi^e \sin\Phi^e \\ \cos\Phi^e \sin\Phi^e & \sin^2\Phi^e & -\cos\Phi^e \sin\Phi^e & -\sin^2\Phi^e \\ -\cos^2\Phi^e & -\cos\Phi^e \sin\Phi^e & \cos^2\Phi^e & \cos\Phi^e \sin\Phi^e \\ -\cos\Phi^e \sin\Phi^e & -\sin^2\Phi^e & \cos\Phi^e \sin\Phi^e & \sin^2\Phi^e \end{array}\right]. \tag{B.12}$$

Vyřešením soustavy $K^{ge} \cdot r^{ge} = F^{ge}$ jsou získány neznámé posuny a reakce obdobně jako u vztahu B.6.

B.5 Vnitřní osové síly

Obdobně jako ve vztahu B.2 lze získat vyjádření osové síly z $N_x = \sigma_x \cdot A$

$$N_x = \frac{EA}{L} \cdot (u_{2x}^{le} - u_{1x}^{le}) \tag{B.13}$$

a je-li

$$u^{le} = [u_{1x}^{le} \quad u_{1y}^{le} \quad u_{2x}^{le} \quad u_{2y}^{le}]^T, \text{ pak} \tag{B.14}$$

$$N_x = \frac{EA}{L} \cdot [-1 \quad 0 \quad 1 \quad 0] \cdot u^{le}. \tag{B.15}$$

Transformací z lokálního do globálního souřadného systému $u^{le} = T^e \cdot u^{ge}$ pak lze obdržet

$$N_x = \frac{EA}{L}[-cos(\Phi^e) \quad -sin(\Phi^e) \quad cos(\Phi^e) \quad sin(\Phi^e)] \cdot u^{ge}. \tag{B.16}$$

Příloha C

Kód pro 10-prutovou konstrukci pro kompilaci do MEX souboru

```
1  //==========================================================================
2  // Jméno       : main.cpp
3  // Autor       : Adéla Pospíšilová
4  // Verze       : 0001
5  // Popis       : Skript pro výpočet posunů a napětí na konstrukci pro 10-prutovou
6  //               příhradovou konstrukci s plnou maticí tuhosti. K řešení soustavy
7  //               rovnic se využívá řešiče pana doc. Kruise. Skript je připraven
8  //               pro kompilaci do MEX souboru.
9  //==========================================================================
10
11 #include <stdio.h>
12 #include <math.h>
13 #include "mex.h"
14
15 static neq =8;   // počet nenulových posunů
16
17 static void reseni_rovnic(double *a,double *y,double *x,long n,long m,long as)
18
19   /*    funkce provádí řešení maticové rovnice A.X=Y
20         lze ji tedy bez problému použít na výpočet inverzní matice
21         matice A je plná
22
23         A(n,n),X(n,m),Y(n,m)
24         as - rozhodovací konstanta
25         as=1 - pivot se hledá jen v případě, že A(i,i)=0
26         as=2 - pivot se hledá pokaždé
27
28         testováno 24.7.1996 pomoci programu test_fin_res.c v /u/jk/TESTY/METODY
29         procedura dává stejné výsledky jako ldl,ldlkon,ldlblok,congrad_sky
30
31         **** otestováno ****
32         */
33 {
34   long    i,j,k,ac,acr,acc=0,aca,aca1,acx,acy,acy1,aci,acj;
35   long    *av;
36   double  s,g;
37
38   av = (long*) calloc (n,sizeof(long));
39
```

```
40  /*************************************************************************/
41  /*  nastavení hodnot vektoru, který udává pořadí jednotlivých neznámých  */
42  /*************************************************************************/
43  for (i=0;i<n;i++){
44     av[i]=i;
45  }
46
47  for (i=0;i<n-1;i++){
48     acr=i;  acc=i;
49     if (as==1){
50      /*****************************************/
51      /*  pivot se hledá jen pokud je A(i,i)=0  */
52      /*****************************************/
53      if (fabs(a[i*n+i])<1.0e-30){
54        /*  vyber pivota  */
55        s=0.0;
56        /*  smycka pres radky  */
57        for (j=i;j<n;j++){
58          aca=j*n+i;
59          /*  smycka pres sloupce  */
60          for (k=i;k<n;k++){
61            if (s<fabs(a[aca])){
62              s=fabs(a[aca]);  acr=j;  acc=k;
63            }
64            aca++;
65          }
66        }
67        if (s==0.0){
68          printf ("\n Singularni matice v procedure gemp (krok %ld).\n",i);
69          abort ();
70        }
71      }
72    }
73    if (as==2){
74      /****************************/
75      /*  pivot se hledá pokaždé  */
76      /****************************/
77      s=0.0;
78      /*  smycka pres radky  */
79      for (j=i;j<n;j++){
80        aca=j*n+i;
81        /*  smycka pres sloupce  */
82        for (k=i;k<n;k++){
83          if (s<fabs(a[aca])){
84            s=fabs(a[aca]);  acr=j;  acc=k;
85          }
86          aca++;
87        }
88      }
89      if (s==0.0){
90        printf ("\n Singularni matice v procedure reseni_rovnic(krok %ld). \n",i);
91        abort ();
92      }
93    }
94
95    /******************/
96    /*  výměna řádků  */
97    /******************/
```

```
98      if (acr!=i){
99        aca=i*n+i;  aca1=acr*n+i;
100       for (j=i;j<n;j++){
101         s=a[aca];
102         a[aca]=a[aca1];
103         a[aca1]=s;
104         aca++;  aca1++;
105       }
106       acy=i*m;  acy1=acr*m;
107       for (j=0;j<m;j++){
108         s=y[acy];
109         y[acy]=y[acy1];
110         y[acy1]=s;
111         acy++;  acy1++;
112       }
113     }
114     /********************/
115     /*  výměna sloupců  */
116     /********************/
117     if (acc!=i){
118       ac=av[i];
119       av[i]=av[acc];
120       av[acc]=ac;
121
122       aca=i;  aca1=acc;
123       for (j=0;j<n;j++){
124         s=a[aca];
125         a[aca]=a[aca1];
126         a[aca1]=s;
127         aca+=n;  aca1+=n;
128       }
129     }
130     /***************/
131     /*  eliminace  */
132     /***************/
133
134     for (j=i+1;j<n;j++){
135       acj=j*n+i;  aci=i*n+i;
136       s=a[acj]/a[aci];
137       /*  modifikace matice A  */
138       for (k=i;k<n;k++){
139         a[acj]-=s*a[aci];
140         acj++;  aci++;
141       }
142       acj=j*m;  aci=i*m;
143       /*  modifikace matice pravych stran Y  */
144       for (k=0;k<m;k++){
145         y[acj]-=s*y[aci];
146         acj++;  aci++;
147       }
148     }
149   }
150
151   /*****************/
152   /*  zpětný chod  */
153   /*****************/
154
155   for (i=n-1;i>-1;i--){
```

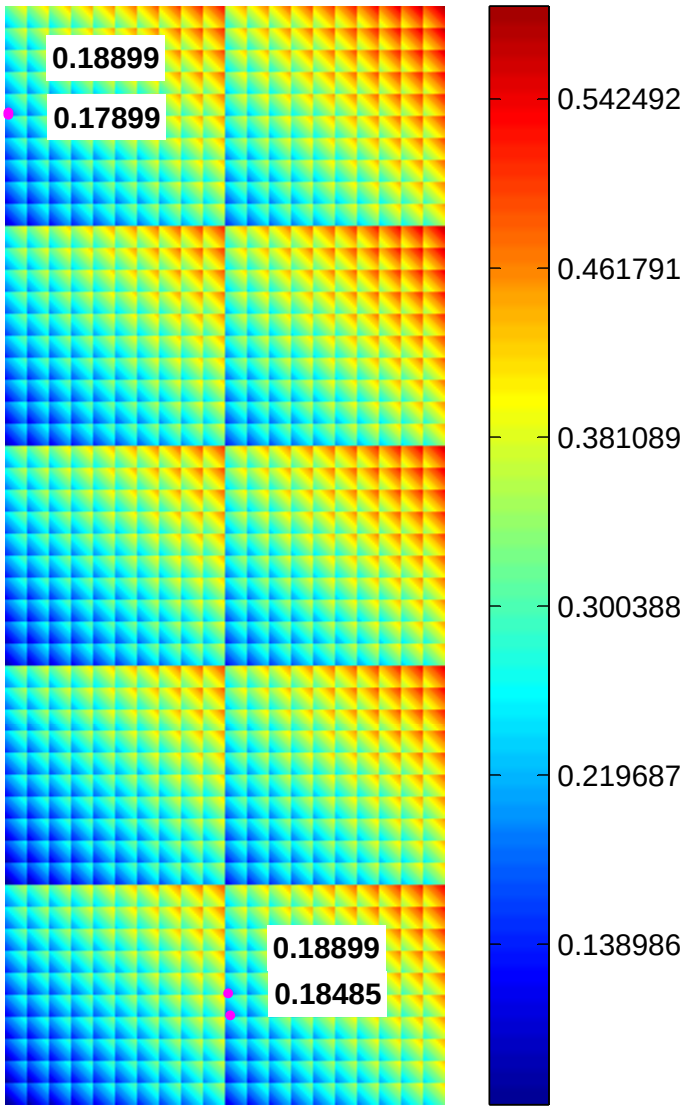
```
156     g=a[i*n+i];  acx=i*m;
157     for (j=0;j<m;j++){
158       s=0.0;  aca=i*n+i+1;  acy=(i+1)*m+j;
159       for (k=i+1;k<n;k++){
160         s+=a[aca]*x[acy];
161         aca++;  acy+=m;
162       }
163       x[acx]=(y[acx]-s)/g;
164       acx++;
165     }
166   }
167
168   /***********************************/
169   /*  přerovnání do původního stavu  */
170   /***********************************/
171   for (i=0;i<n;i++){
172     if (av[i]!=i){
173       for (j=i;j<n;j++){
174         if (av[j]==i){
175           acc=j;  break;
176         }
177       }
178
179       ac=av[i];
180       av[i]=av[acc];
181       av[acc]=ac;
182
183       aca=i*m;  aca1=acc*m;
184       for (j=0;j<m;j++){
185         s=x[aca];
186         x[aca]=x[aca1];
187         x[aca1]=s;
188         aca++;  aca1++;
189       }
190     }
191   }
192
193   free (av);
194 }
195
196 static void compute ( double A[], double OUT[] )
197 /*  funkce řešící posuny, napětí a váhu konstrukce pomocí MKP
198        A      ... jednorozměrné pole s plochami příčných řezů
199        OUT[0] ... maximální posun
200        OUT[1] ... maximální napětí
201        OUT[2] ... váha konstrukce  */
202 {
203   int i;
204   double k1, k2, k3, k4, k5, k6, k7, k8, k9, k10;
205   double f[8];      // počet rovnic
206   double K[8*8];    // počet rovnic * počet rovnic
207   double u[8];      // počet rovnic
208   double S[10] ;    // počet ploch příčných řezů
209   double maxw = 0 ;
210   double maxs = 0 ;
211
212   k1 = 250.0/9.0*A[0];
213   k2 = 250.0/9.0*A[1];
```

```
214    k3 = 250.0/9.0*A[2];
215    k4 = 250.0/9.0*A[3];
216    k5 = 250.0/9.0*A[4];
217    k6 = 250.0/9.0*A[5];
218    k7 = 0.17592186044416E18/0.8956478914489369E16*A[6];
219    k8 = 0.17592186044416E18/0.8956478914489369E16*A[7];
220    k9 = 0.17592186044416E18/0.8956478914489369E16*A[8];
221    k10 = 0.17592186044416E18/0.8956478914489369E16*A[9];
222
223    for ( i=0 ; i<(neq*neq) ; i++ ) {
224       K[i]=0.0;
225    }
226
227    K[0*8+0] = k10*(1.0/2.0)+k2;  K[0*8+1] = k10*(1.0/2.0);  K[0*8+4] = -k2;
228    K[0*8+6] = k10*(-1.0/2.0);  K[0*8+7] = k10*(-1.0/2.0);  K[1*8+0] = k10*(1.0/2.0);
229    K[1*8+1] = k10*(1.0/2.0)+k6;  K[1*8+3] = -k6;  K[1*8+6] = k10*(-1.0/2.0);
230    K[1*8+7] = k10*(-1.0/2.0);  K[2*8+2] = k4+k9*(1.0/2.0);  K[2*8+3] = k9*(-1.0/2.0);
231    K[2*8+4] = k9*(-1.0/2.0);  K[2*8+5] = k9*(1.0/2.0);  K[2*8+6] = -k4;
232    K[3*8+1] = -k6;  K[3*8+2] = k9*(-1.0/2.0);  K[3*8+3] = k6+k9*(1.0/2.0);
233    K[3*8+4] = k9*(1.0/2.0);  K[3*8+5] = k9*(-1.0/2.0);  K[4*8+0] = -k2;
234    K[4*8+2] = k9*(-1.0/2.0);  K[4*8+3] = k9*(1.0/2.0);
235    K[4*8+4] = k1+k2+k8*(1.0/2.0)+k9*(1.0/2.0);  K[4*8+5] = k8*(1.0/2.0)-k9*(1.0/2.0);
236    K[5*8+2] = k9*(1.0/2.0);  K[5*8+3] = k9*(-1.0/2.0);
237    K[5*8+4] = k8*(1.0/2.0)-k9*(1.0/2.0);  K[5*8+5] = k5+k8*(1.0/2.0)+k9*(1.0/2.0);
238    K[5*8+7] = -k5;  K[6*8+0] = k10*(-1.0/2.0);  K[6*8+1] = k10*(-1.0/2.0);
239    K[6*8+2] = -k4;  K[6*8+6] = k10*(1.0/2.0)+k3+k4+k7*(1.0/2.0);
240    K[6*8+7] = k10*(1.0/2.0)-k7*(1.0/2.0);  K[7*8+0] = k10*(-1.0/2.0);
241    K[7*8+1] = k10*(-1.0/2.0);  K[7*8+5] = -k5;
242    K[7*8+6] = k10*(1.0/2.0)-k7*(1.0/2.0);  K[7*8+7] = k10*(1.0/2.0)+k5+k7*(1.0/2.0);
243
244
245    f[0]=0;  f[1]=0;  f[2]=0;  f[3]=-100;  f[4]=0;  f[5]=0;  f[6]=0;  f[7]=-100;
246
247    reseni_rovnic(K,f,u,neq,1,1);
248
249    S[0] = fabs(k1*u[4]/A[0]);  S[1] = fabs(k2*(u[0]-u[4])/A[1]);
250    S[2] = fabs(k3*u[6]/A[2]);  S[3] = fabs(k4*(u[2]-u[6])/A[3]);
251    S[4] = fabs(k5*(u[5]-u[7])/A[4]);  S[5] = fabs(k6*(u[1]-u[3])/A[5]);
252    S[6] = fabs(k7*(u[6]*sqrt(2.0)/2.0-u[7]*sqrt(2.0)/2.0)/A[6]);
253    S[7] = fabs(k8*(u[4]*sqrt(2.0)/2.0+u[5]*sqrt(2.0)/2.0)/A[7]);
254    S[8] = fabs(k9*((u[2]/2.0-u[4]/2.0)*sqrt(2.0)-(u[3]/2.0-u[5]/2.0)*sqrt(2.0))/A[8]);
255    S[9] = fabs(k10*((u[0]/2.0-u[6]/2.0)*sqrt(2.0)+(u[1]/2.0-u[7]/2.0)*sqrt(2.0))/A[9]);
256
257    for ( i=0 ; i<8 ; i++ )
258       if ( fabs(u[i]) > maxw ) maxw = fabs(u[i]) ;
259    OUT[0] = maxw;
260
261    for ( i=0 ; i<10 ; i++ )
262      if ( S[i] > maxs ) maxs = S[i] ;
263
264    OUT[1] = maxs;
265    OUT[2] = 36.0*A[0]+36.0*A[1]+36.0*A[2]+36.0*A[3]+36.0*A[4]+36.0*A[5]
266    + 0.8956478914489369E16/175921860444160.0*A[6]
267    + 0.8956478914489369E16/175921860444160.0*A[7]
268    + 0.8956478914489369E16/175921860444160.0*A[8]
269    + 0.8956478914489369E16/175921860444160.0*A[9];
270 }
271
```

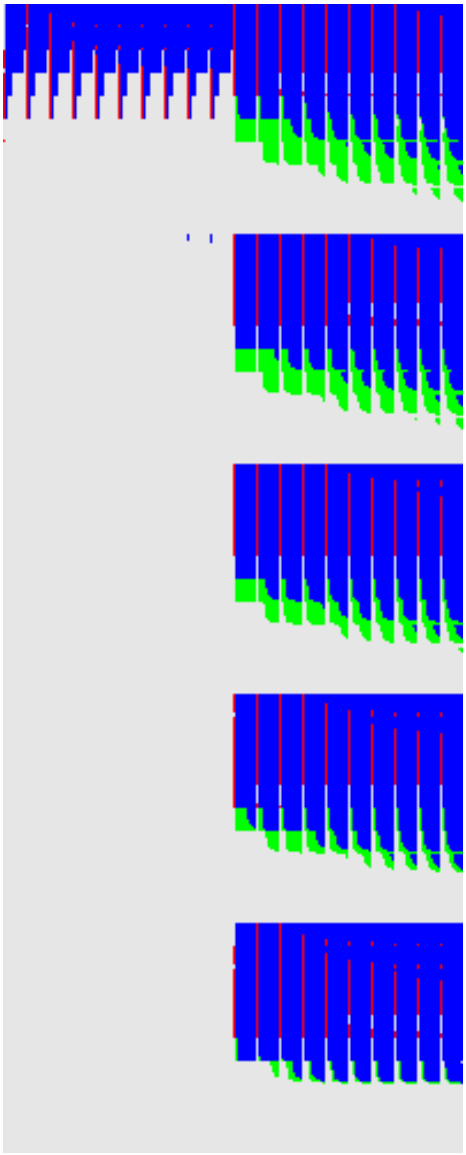
```
272 void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray*prhs[] )
273 {
274   /* alokace vstupů, výstupů a pomocných polí */
275   double *K, *f, *u, *S, *A, *OUT;
276 
277   /* vstupy */
278   A = mxGetPr(prhs[0]);    // vytvořím si pointer na data ve vstupu A
279 
280   /* výstup */
281   plhs[0] = mxCreateDoubleMatrix(3,1,mxREAL);
282   OUT = mxGetPr(plhs[0]);
283 
284   compute(A,OUT);
285   return;
286 }
```

Příloha D

Hypergrafy

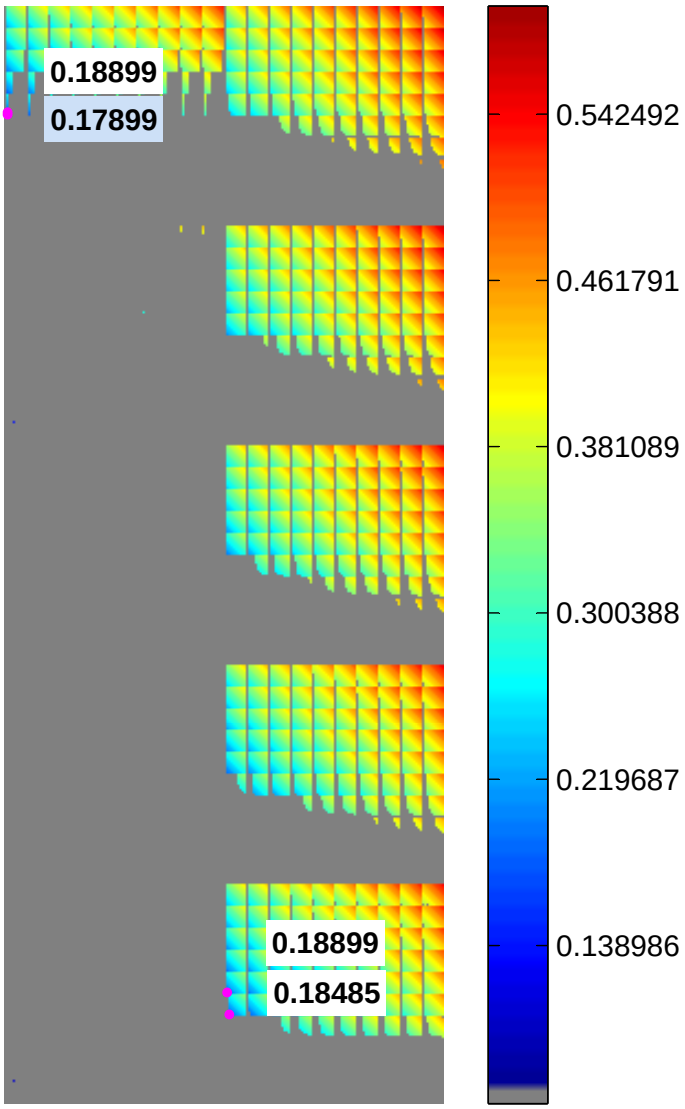


Obrázek D.1: Hypergraf pětiprutové konstrukce s vykreslením všech hmotností pro celou úlohu a zakreslením 4 nejlepších řešení (legenda v lb), liché proměnné jsou svisle, sudé jsou vodorovně, 1. proměnná tvoří dva sloupce

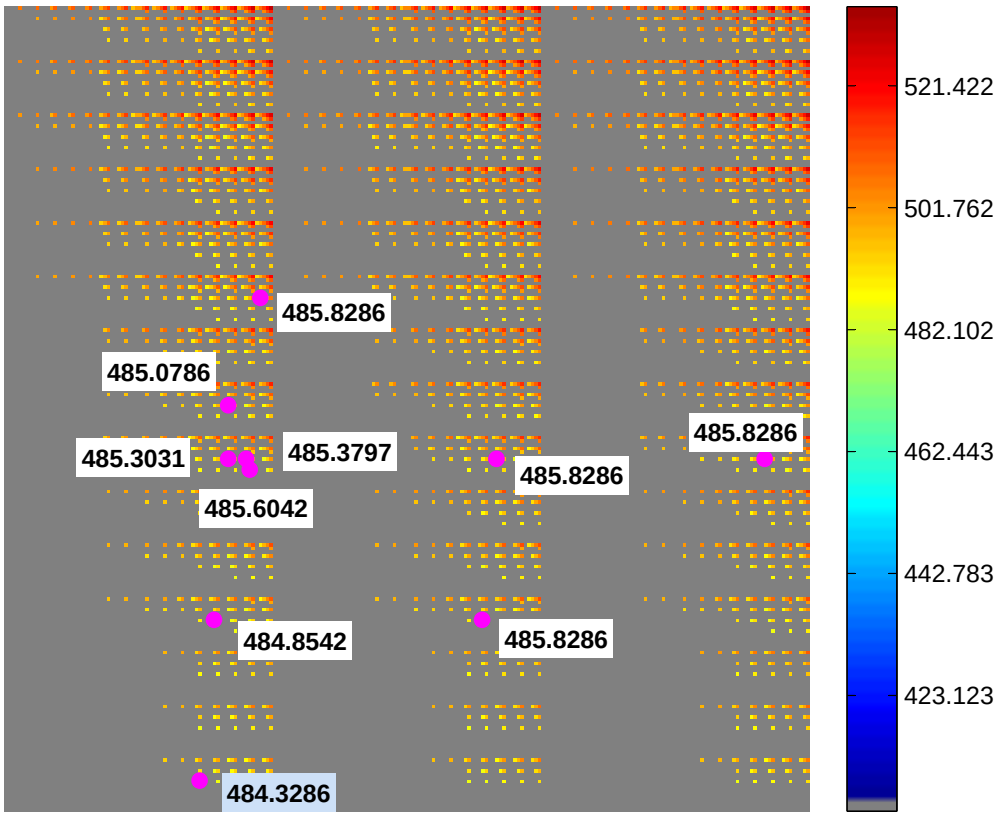


Obrázek D.2: Hypergraf pětiprutové konstrukce s vykreslením splnění omezujících podmínek pro celou úlohu (šedá - nevyhovuje žádná podmínka, zelená - vyhovuje pouze napětí, červená - vyhovuje pouze posun, modrá - omezující podmínky splněny), liché proměnné jsou svisle, sudé jsou vodorovně, , 1. proměnná tvoří dva sloupce

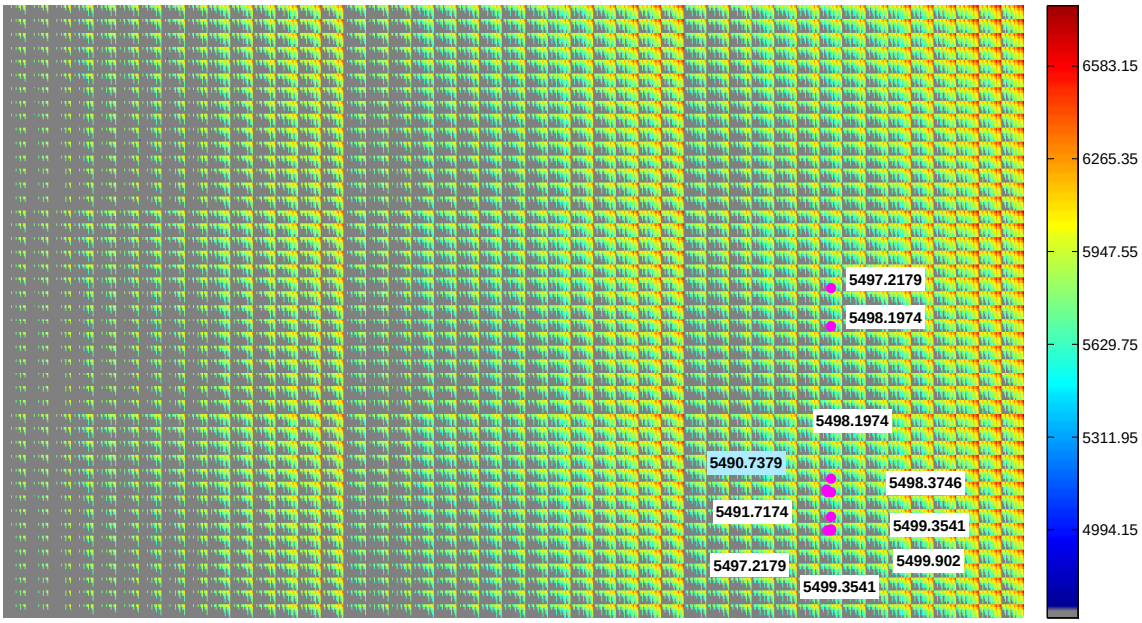
Na následujících obrázcích jsou zobrazeny hypergrafy pro konstrukce popsané v kapitole 2. Na obrázku D.1 jsou znázorněna všechna zadání s hmotnostmi v librách vypsaná metodou hrubé síly. Na obrázku D.2 jsou znázorněny vyhovující a nevyhovující podmínky pro všechna zadání. Na obrázku D.3 je znázorněn průnik předchozích grafů, tedy vykreslení všech vyhovujících zadání omezujícím podmínkám. Pro ostatní konstrukce jsou zobrazeny pouze tyto průniky, zadání jsou uvedena v popisku obrázku včetně maximálních změn profilů.



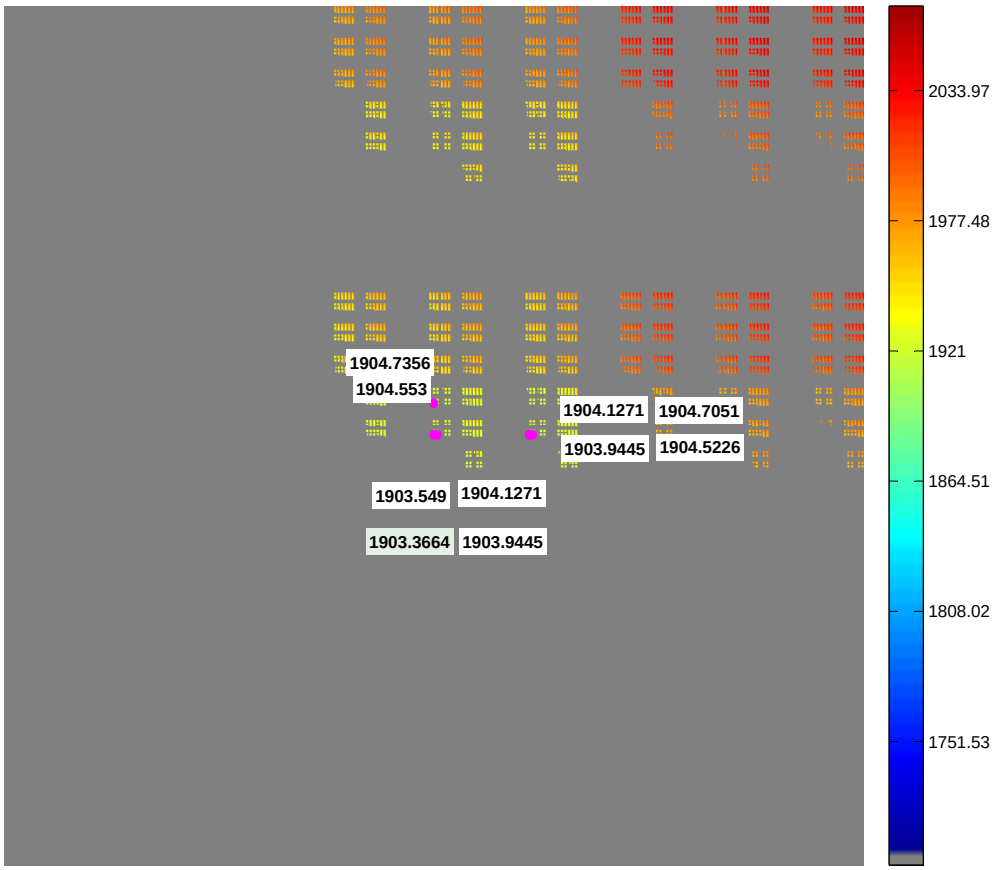
Obrázek D.3: Hypergraf pětiprutové konstrukce s vykreslením hmotností u splněných omezujících podmínek a zakreslením 4 nejlepších řešení (nevyhovující šedě, legenda v lb), liché proměnné jsou svisle, sudé jsou vodorovně, 1. proměnná tvoří dva sloupce



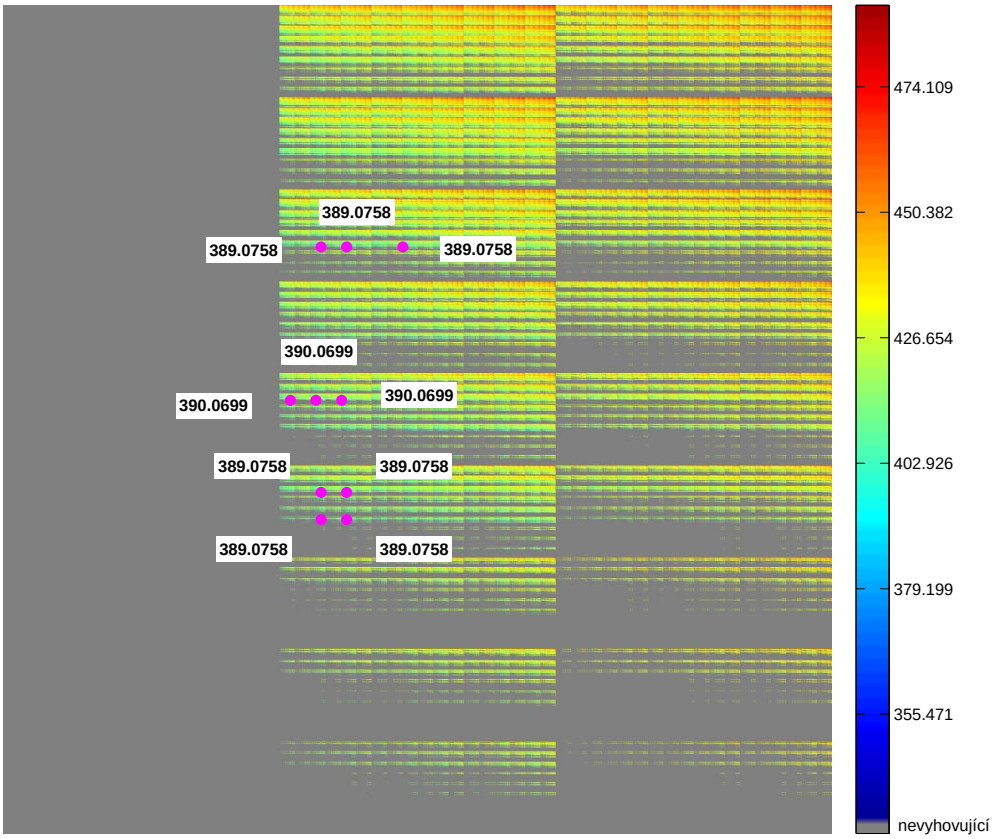
Obrázek D.4: Hypergraf 25-prutové konstrukce s vykreslením hmotností u splněných omezujících podmínek a zakreslením 10 nejlepších řešení, změna zadání maximálně o dva profily, výchozí řešení viz Kripka, tabulka 6.5 (nevyhovující šedě, legenda v lb), liché proměnné jsou vodorovně, sudé jsou svisle



Obrázek D.5: Hypergraf 10-prutové konstrukce s vykreslením hmotností u splněných omezujících podmínek a zakreslením 10 nejlepších řešení, změna zadání maximálně o dva profily, výchozí řešení viz Cai, tabulka 6.1 (nevyhovující šedě, legenda v lb), liché proměnné jsou vodorovně, sudé jsou svisle



Obrázek D.6: Hypergraf 52-prutové konstrukce s vykreslením hmotností u splněných omezujících podmínek a zakreslením 10 nejlepších řešení, změna zadání maximálně o jeden profil, výchozí řešení viz Giger, tabulka 6.7 (nevyhovující šedě, legenda v lb), liché proměnné jsou vodorovně, sudé jsou svisle



Obrázek D.7: Hypergraf 72-prutové konstrukce s vykreslením hmotností u splněných omezujících podmínek a zakreslením 10 nejlepších řešení, změna zadání maximálně o jeden profil, výchozí řešení viz Wu, tabulka 6.8 (nevyhovující šedě, legenda v lb), liché proměnné jsou vodorovně, sudé jsou svisle

Příloha E

Význam zkratek jednotlivých metod

zkratka	metoda
ABC	Artificial Bee Colony Algorithm
AC	Ant Colony Optimization
CA	Combined Approach
CMS	Convex Model Superposition
DCOC	Discrete version of continuum-type optimality criteria method
DE	Differential Evolution
DHPSACO	Discrete heuristic particle swarm ant colony optimization
ES	Evolution Strategies
GA	Genetic Algorithm
GA-APM	Genetic Algorithm - adaptive penalty method
GA-CPM	Genetic Algorithm - constant penalty parameters
GEO	Generalized Extremal Optimization
GGP	General Geometric Programming
HOC	Hybrid optimality criteria
HPSACO	Heuristic particle swarm ant colony optimization
HPSO	Heuristic Particle Swarm Optimizer
HS	Harmony Search Algorithm
MABC	Modified Artificial Bee Colony Algorithm
MDNLP	Mix-Discrete NonLinear Programming
MOP	Multi-Objective Problem
PSO	Particle Swarm Optimizer
PSOPC	Particle Swarm Optimizer with Passice Congregation
SA	Simulated Annealing
SGA	Simple Genetic Algorithm
SOC	Simple optimality criteria
SQP	sequential quadratic programming
TPO	Two-Phase Optimization Method
TS	Tabu Search

Tabulka E.1: Soupis jednotlivých metod včetně použitých zkratek

Příloha F

Obsah přiloženého CD

Na přiloženém CD jsou k dispozici tyto soubory.

- Řešiče soustav lineárních rovnic
  - Zdrojové kódy pro MATLAB
    - Sestavení matice tuhosti přímo, zpětné lomítko
    - Plná matice tuhosti
      - Zpětné lomítko
      - LU faktorizace
      - Choleského dekompozice
      - LDL^T rozklad
      - Metoda sdružených gradientů
    - Řídká matice tuhosti
      - Zpětné lomítko
      - LU faktorizace
      - Choleského dekompozice
      - LDL^T rozklad
      - Metoda sdružených gradientů
    - MEX soubory
      - Řešič
      - MKP
  - Zdrojové kódy v C/C++
    - LDL^T rozklad
    - LAPACK++
    - FEMCad
- Metoda hrubé síly (zdrojové kódy pro MATLAB)
- Metoda větví a mezí
  - Zdrojové kódy pro MATLAB
  - Zdrojové kódy v C/C++