UNIVERZITA PARDUBICE
DOPRAVNÍ FAKULTA JANA PERNERA

OPTIMALIZACE VYUŽITÍ LOŽNÉHO PROSTORU

Autor: **Ing. Jaroslav Koziol**
Školitel: **Prof. Ing. Václav Cempírek, Ph.D.**

Disertační práce

2008

UNIVERSITY OF PARDUBICE
JAN PERNER TRANSPORT FACULTY

OPTIMISATION OF USAGE OF STORAGE SPACE

Author: **Ing. Jaroslav Koziol**
Supervisor: **Prof. Ing. Václav Cempírek, Ph.D.**

Disertation

2008

OBSAH

ÚVOD

Optimálně fungující doprava je jedním z rozhodujících činitelů pro uplatnění logistických procesů a dosažení z toho plynoucích efektů v řízení toků zboží. Z logistického hlediska se optimálně fungující dopravou rozumí doprava nejen rychlá, spolehlivá a bezpečná, ale i levná. Dodržení těchto vlastností nezávisí pouze na samotné přepravě zboží a k tomu použité technologii, ale také na navazujících článcích logistického řetězce. Z těchto článků je to především nakládka a zajištění zboží, co značně ovlivňuje kvalitu a cenu samotné přepravy. Cena může být rovněž ovlivněna způsobem rozložení zboží mezi vhodné dopravní prostředky.

Práce se zabývá optimalizací využití ložného prostoru. Nutno upozornit, že se nejedná pouze o maximální využití rozměrové kapacity dopravního prostředku, jak by se mohlo na první pohled zdát. Do řešení praktického problému totiž vstupuje takové množství rozličných omezujících podmínek, že výsledek nakládání se může od této představy značně lišit.

Naložené zboží se téměř vždy někam přemísťuje a tento proces přemístění má nějaké vlastnosti a specifika, které mají vliv na bezpečnost naloženého zboží. Tyto vlastnosti vycházejí z vlastností ložného prostoru – nejčastěji dopravního prostředku. Každý dopravní prostředek má kromě omezeného ložného prostoru také nějakou maximální nosnost. Toto jsou takzvané kapacitní omezující podmínky. Je však třeba dodržet i další specifické podmínky, například určitý poměr rozložení hmotnosti na ploše dopravního prostředku či na nápravy dopravního prostředku a mnoho dalších. Dále při přepravě samotné působí na naložené zboží mnoho dynamických vlivů, které mohou mít na naložené zboží vliv, například při projíždění dopravního prostředku zatáčkou, při rozjezdu a brzdění, při případném nárazu.

S omezením rozměrů dopravního prostředku se člověk při manuálním řešení vyrovná poměrně snadno. Horší situace nastává například při řešení hmotnostního omezení, které se při tvorbě ložného plánu vnímána spíše abstraktně. Lidský mozek potom přirozeně tato omezení zanedbává a výsledné řešení nemusí být správné. Závažnost tohoto problému je v přímé úměře s rostoucím rozměrem úlohy a množstvím omezujících podmínek. Naopak počítače jsou schopny respektovat sebemenší omezení a umí pracovat s obrovským objemem dat. Chybí jim však to, co je lidskému mozku vlastní – nedokáží se učit, pozorovat a poznávat souvislosti, „vidět“, rozpoznávat a rozlišovat neznámé objekty, generalizovat, logicky myslet. I když i toto tvrzení nemusí být zcela pravdivé…

V úvodní kapitole disertační práce se zabývám analýzou současného vědeckého poznání v oblasti optimalizace využití ložného prostoru. Stručný přehled některých softwarových aplikací nabízí pohled na to, jak se s tímto problémem vypořádává praxe.

Následující kapitola popisuje model úlohy tak, jak je navržen pro vlastní řešení, popisuje typ úlohy, definuje omezující podmínky a možná kritéria pro posuzování kvality řešení. Součástí je podrobná specifikace datových struktur, která je potřebná pro další řešení.

Zcela nový přístup k řešení, jakožto i podrobný popis postupu řešení úlohy nakládání je podrobně popsán v kapitole 4. Postup řešení podchycuje nově sestavený samoučcí se meta-algoritmus, který obsahuje upravené či zcela nově navržené dílčí metody, algoritmy a mechanismy pro jejich spolupráci.

Závěrečná 5. kapitola se zabývá posouzením dosažených výsledků, porovnáním kvality získaných řešení. Dále stručně představuje softwarové aplikace, které se pro řešení používají. Následuje celkové zhodnocení výsledků disertační práce.

Jeden z problémů, na které jsem narazil při řešení této problematiky, je otázka české terminologie. Ne ke všem anglickým termínům existují vhodné české ekvivalenty. Například ložný prostor dopravního nebo přepravního prostředku nazývám obecně „kontejnerem“ (v anglické terminologii je to „Bin“) a pro nakládané zboží používám obecné pojmenování „objekt“. Další podobné problémy vyvstávají u pojmenování metod a algoritmů.

Dalším problémem je, že vzhledem k obsáhlosti zpracovávaných dat většinou nelze v textu disertační práce prezentovat reálné příklady a ukázky. Principy některých algoritmů a výpočetních postupů jsou proto ilustrovány na zjednodušený příkladech.

V práci jsem využil své osmileté zkušenosti s problematikou optimalizace nakládání a ještě delší zkušenosti programátorské praxe. Přesto bych rád poděkoval všem, se kterými jsem mohl spolupracovat a kteří ovlivnili můj pohled na tuto problematiku.

1 Analýza současného stavu poznání

1.1 Teoretický základ problému

Problémy vztahující se ke kapacitnímu omezení teoreticky řeší diskrétní matematika množinou úloh o balení. V odborné práci používám výstižnější anglické pojmenování Bin Packing (BP). Obecně se nejedná o jediný problém, ale o celou skupinu zahrnující poměrně širokou oblast diskrétní matematiky (14). Praktické použití a výskyty tohoto problému jsou popsány níže v kapitole 1.2.

Cílem úlohy je určit, jak umístit danou množinu nakládaných objektů do minimálního množství ložných jednotek (dále je používán pojem „kontejner“) nebo jak umístit do daného počtu kontejnerů maximální množství objektů. Kontejnerem chápeme obecně jiný objekt, který má předem určenou a konstantní kapacitu (v angličtině „bin“). Úkolem je nalézt rozdělení množiny objektů a jejich přiřazení ke kontejnerům, do kterých mají být uloženy při dodržení stanovených omezujících podmínek a minimalizaci či maximalizaci účelové funkce.

I přes dynamický rozvoj a nástup výpočetní techniky do řešení takovýchto problémů, který nastal zhruba před 30 lety, zůstal Bin Packing problém aktuální a „klasicky“ obtížný až do dnešních dnů. Diskrétní matematika se tímto problémem zabývá již po desítky let, ale nikdo doposud nepředstavil algoritmus, pomocí kterého lze získat optimální řešení v „rozumném“ čase. Jak bude dokázáno níže, ani stále rostoucí výkonnost a rychlost počítačů při řešení takovéhoto druhu problému nepomůže. Nicméně doposud nikdo ani nevyvrátil možnost existence takového algoritmu. Proto teoretici zařazují tento problém do skupiny „neřešitelných“ problémů, v matematice známých jako NP-těžké a NP-úplné. Jako většina NP-těžkých problémů je Bin Packing ve stálém zájmu vědeckých pracovníků a počítačových odborníků. Nejedná se však pouze o akademicky populární problém, ale především o problém praktický. Zatímco však teoretici zkoumají, přemýšlí a testují, praktický svět spoléhá na řešení, které sice není vždy optimální, ale pro reálnou potřebu bývá víceméně dostačující.

V mnoha odvětvích, od plánování a rozvrhování televizních programů po průmysl, dopravu a logistiku, se k řešení kapacitních problémů některá z úloh Bin Packingu často používá. V mnoha případech, kdy je výpočetní technika schopna nalézt uspokojivé řešení v dostupném čase, toto řešení praktickým potřebám vyhovuje. Z tohoto důvodu jsou vyvíjeny heuristické algoritmy na řešení BP problému. Nicméně, vždy je zde prostor pro další zlepšení či zrychlení výpočtu, zahrnutí dalších omezujících podmínek, které přiblíží model úlohy blíže reálnému problému.

1.1.1 Analýza výpočetní složitosti

Bin Packing patří do skupiny NP-těžkých úloh (17), tedy úloh, kde nelze najít řešení v polynomiálním čase. Pro jejich řešení tedy platí všechny předpoklady a důsledky, které z toho vyplývají. Obecně se jedná o NP-těžké kombinatorické úlohy,

což je skupina velmi obtížně naprogramovatelných úloh, zvláště při daném stupni obecnosti.

Pro lidský mozek, tedy lidskou inteligenci, se úloha zdá zřejmá. Při malém počtu vstupů a omezujících podmínek je i poměrně snadno řešitelná. Při větším rozsahu úlohy však člověk není schopen vnímat systematicky všechny vstupy a okrajové podmínky. V těchto situacích se mozek přirozeně snaží ulehčit si práci a má snahu vynechávat ty vstupy a omezující podmínky, které se mu jeví méně podstatné. Naopak úplné řešení těchto úloh pomocí výpočetní techniky je již na pokraji umělé inteligence.

Pro ilustraci následuje několik příkladů rozdílů mezi algoritmy pracujícími v polynomiálně omezeném čase a algoritmy s vyšší složitostí, například exponenciálními. Následující tabulka Tabulka 1.1 Časová náročnost dle výpočetní složitosti algoritmů (11) uvádí čas potřebný ke zpracování vstupních dat velikosti n, jestliže počet operací, které je nutno provést, je udán funkcí $f(n)$. Dále se předpokládá, že provedení jedné operace trvá jednu mikrosekundu.

Tabulka 1.1 Časová náročnost dle výpočetní složitosti algoritmů (11)

f(n) \ n	20	40	60	80	100	200	500	1000
n	20 μs	40 μs	60 μs	80 μs	100 μs	200 μs	500 μs	1000 μs
n*log n	86 μs	0,2 ms	0,35 ms	0,5 ms	0,7 ms	1,5 ms	4,5 ms	10 ms
n^2	0,4 ms	1,6 ms	3,6 ms	6,4 ms	10 ms	40 ms	0,25 s	1 s
n^3	8 ms	64 ms	0,22 s	0,5 s	1 s	8 s	125 s	17 min
n^4	0,16 s	2,56 s	13 s	41 s	100 s	27 min	17 h	11,6 dní
2^n	1 s	11,7 dní	36600 let	$3{,}6 \cdot 10^9$ let	-	-	-	-
n!	77000 let	-	-	-	-	-	-	-

Z tabulky Tabulka 1.1 je zřejmé, že doba výpočtu rozsáhlejších úloh se neúměrně prodlužuje již při polynomiální výpočetní složitosti a vyšším stupni polynomu.

Ještě patrnější rozdíl mezi algoritmy o různé výpočetní složitosti je z tabulky Tabulka 1.2 ukazující zvětšení rozsahu zpracovatelnosti úloh, které odpovídá zvětšení výpočetní rychlosti použitého počítače v násobcích 10x, 100x 1000x. Je použit předpoklad, že původně bylo možné v daném časovém limitu zpracovat vstupní data o velikosti $n = 100$.

Tabulka 1.2 Vliv zvýšení rychlosti výpočtu na dobu výpočtu (11)

f(n)	1x	10x	100x	1000x
N	100	1 000	10 000	100 000
n * log n	100	702	5 362	43 150
n^2	100	316	1 000	3 162
n^3	100	215	464	1 000
n^4	100	177	316	562
2^n	100	103	106	109
n!	100	100	100	101

Z tabulky Tabulka 1.2 je opět zřejmé, že pro exponenciální algoritmy je typická existence mezní velikosti vstupních dat, nad níž je úloha neřešitelná i v případě, že by došlo ke zvýšení rychlosti počítače o několik řádů.

Exaktní řešení lze získat pouze metodou takzvané „hrubé síly“, tedy procházením a porovnáním všech možných variant naložení. To však není vzhledem k rozsahu úlohy vždy možné. S větším počtem přepravovaných kusů roste počet možností jejich naložení velmi rychle.

Příklad:

Jako příklad z řešené oblasti je uveden následující rozbor výpočetní složitosti při nakládání dvou objektů (8). Samotný kvádr lze při nejobecněji zadaných vlastnostech (otáčení kolem vertikální i horizontální osy) naložit v 6 různých polohách. Naložení dvou obecných kvádrů již dává 36 kombinací týkajících se jen nezávisle jejich polohy. Dalších 6 možností je dáno jejich vzájemnou polohou. Pro dva tyto objekty je to tedy celkem 216 kombinací a to ještě nebyly brány v úvahu různé druhy kontejnerů, do nichž je možno tyto dva objekty naložit.

Z tohoto příkladu je patrné, že v úlohách s několika stovkami objektů není možné použití metody hrubé síly jakožto jediné známé metody obecně poskytující exaktní řešení, protože nelze získat požadované řešení v reálném čase.

1.1.2 Členění Bin packing úloh

Jak bylo již zmíněno v úvodu, existuje mnoho variant úloh založených na Bin Packing problému. Toto členění je nezbytné pro bližší specifikaci problému a následné použití specializovaných heuristických metod. Základní rozdělení úloh je podle:

a) Dimenzí, ve kterých je problém řešen.

   Snadno představitelné je to na příkladu prostorových dimenzí, což i skutečně odpovídá mnoha variantám tohoto problému:

- lineární,
- plošný (2D Bin Packing),
- prostorový (3D Bin Packing).

S každou dimenzí samozřejmě výrazně narůstá složitost řešení úlohy. Největší nárůst složitosti je patrný především mezi lineárním a plošným modelem.

c) Homogennosti objektů:

- homogenní,
- heterogenní.

Pokud mají všechny objekty stejné vlastnosti (homogenní), je to pro řešení úlohy značné zjednodušení. V klasickém Bin Packing problému jsou uvažovány objekty s obecně různými parametry (heterogenní).

b) Tvaru nakládaných objektů.

Při teoretickém řešení Bin Packing problému se uvažují objekty, které splňují určitá pravidla. Tak například ve dvou-dimenzionálním BP představují objekty obdélníky, v třídimenzionálním kvádry. V praktických úlohách se však může jednat také o kruhy respektive válce nebo koule, či jiné tvary. Podle tvaru lze objekty rozdělit také na konvexní a nekonvexní.

d) Homogennosti kontejnerů.

Řešení úlohy, kde je k dispozici pouze jeden typ kontejneru, je jednodušší než řešení úlohy s různými kontejnery. Zde je totiž potřeba stanovit pravidla a připravit metody pro výběr vhodného typu kontejneru.

e) Účelové funkce.

Účelová funkce určuje, jakým směrem se má optimalizace ubírat, tedy co se bude optimalizovat. V praktických úlohách existuje celá řada požadavků na kvalitu řešení. Bohužel si tato kritéria často odporují, takže není snadné sestavit vhodnou a obecně použitelnou účelovou funkci.

f) Způsobu vstupu objektů do úlohy:

- Stochasticky – v průběhu řešení je možné přijímat další vstupující objekty, o jejichž vlastnostech není předem nic známo. Obecně se používají takzvané on-line algoritmy, zde tedy on-line Bin Packing.
- Deterministicky - počet a vlastnosti všech objektů jsou předem známy, po zahájení výpočtu již nelze množinu zpracovávaných objektů měnit (off-line Bin Packing).

I když se na první pohled zdá z hlediska vlastního nakládání řešení on-line problému obtížnější, není tomu tak. Řešení off-line totiž musí uvažovat celou

množinu nakládaných objektů, zatímco při on-line problému se při postupném vstupu objektů do úlohy dílčí řešení již nepřepočítává.

g) Přístupem k řešení.

Například problém, jak vměstnat maximální množství objektů do předem určeného kontejneru (nebo stanoveného počtu kontejnerů), vyžaduje jiný přístup než problém, jak naložit všechny objekty do minimálního počtu kontejnerů.

Dále může být problém chápán buď jako optimalizační nebo rozhodovací. V rozhodovacím problému musí být zodpovězena otázka: „Umístí se všechny objekty do kontejneru?“. Tedy jestli pro množinu objektů existuje kapacita, do které by se vešly nebo ne. Oproti tomu optimalizační problém se pokouší minimalizovat potřebný počet kontejnerů nebo minimalizovat jejich nevyužitou kapacitu. Protože řešením rozhodovacího problému je odpověď „ano“ nebo „ne“, je i jeho samotné řešení méně komplikované než řešení optimalizačního problému. V případě klasického Bin Packing problému je rozhodovací problém považován za NP-úplný, zatímco optimalizační za NP-těžký (3) (17).

h) Respektovaných omezujících podmínek.

Omezujících podmínek vstupujících do tohoto problému může být celá řada. Některé již byly naznačeny v úvodu. Obsáhlejší výpis následuje níže v kapitole 1.1.5.

Kombinací všech těchto možností vzniká velmi obsáhlá množina úloh. Protože doposud neexistuje obecný exaktní algoritmus k získání řešení v polynomiálním čase, používají se algoritmy heuristické. Ty však musí být úzce zaměřeny na každý z těchto problémů.

1.1.3 Klasický Bin Packing problém

Pro řádné pochopení problematiky představím nejprve klasický Bin Packing problém. Za klasický Bin Packing problém se považuje jednodimensionální Bin Packing problém (14). Konečná množina objektů charakterizovaných svou jednou veličinou (například velikostí) má být naskládána do jednoho nebo více kontejnerů. Každý takový kontejner může uchovávat libovolnou podmnožinu objektů. Ty ale nesmějí součtem svých charakterizujících veličin překročit kapacitu kontejneru. Objekty jsou do kontejneru vkládány tak, aby zbývající volná kapacita kontejneru byla minimální.

Klasický BP problém nerozlišuje, zda se jedná o problém rozhodovací nebo optimalizační ve smyslu jak bylo uvedeno předchozí kapitole. Objekty jsou již z podstaty úlohy heterogenní, jejich tvar není vzhledem k jednorozměrnosti úlohy podstatný.

1.1.4 Rozšířený Bin Packing problém

Klasický Bin Packing problém se zabývá pouze konvexními a ortogonálními objekty[1] (nelze s nimi otáčet). Do úloh odpovídajících praxi však často nelze zahrnout všechny omezující podmínky. Definice klasického BP problému proto slouží jako výchozí model pro další úpravy. Proto jako jedno z rozšíření klasického Bin Packing problému bude umožněno otáčení objektů kolem jejich vertikální osy.

Dalším rozšířením budeme chápat především navýšení počtu dimenzí problému. Již rozšířením klasického Bin Packing problému o jednu další dimenzi se značně zkomplikuje jeho řešení. Objekty jsou definovány již dvěmi proměnnými veličinami. Problém si lze představit jako umísťování obdélníků na omezenou plochu. Další příklad řešení podobného problému lze spatřit například ve známé hře Tetris.

Třídimensionální BP, jak lze předvídat, řeší umístění objektů, které existují ve třech dimenzích (délka, šířka, výška). Každý objekt, zde tedy kvádr, by se měl umístit do prostorového kontejneru co nejvhodněji. Stejně jako v 2D BP musí být při nakládání zachována orientace objektů. 3D BP si lze tedy představit jako umísťováním menšících krabic do větší.

Pro 3D Bin Packing lze uvažovat jediný kontejner nebo více kontejnerů. Pokud je použit jediný kontejner, tak tento kontejner může být definován tak, že délka a šířka je určena, zatímco výška je neomezená (Open Bin Packing Problem). Tento typ kontejneru umožňuje umísťování objektů dokud není celá množina objektů naložena. Řešení s nekonečnou kapacitou kontejneru obecně vede k efektivnějšímu využití prostoru (17), méně však odpovídá řešení praktických problémů.

Druhý způsob řešení uvažuje více kontejnerů, přičemž každý má již svůj objem předem definovaný. Pokud objem nakládaných objektů přesáhne objem kontejneru, je třeba se rozhodnout, které objekty v kontejneru ponechat a které opět vyjmout. Tento postup je vhodný pro deterministický přístup k řešení a odpovídá na otázku, zda-li mají kontejnery dostatečnou kapacitu k umístění dané množiny objektů.

1.1.5 Omezující podmínky

U všech typů bin packing úloh mohou být do modelu zapracovány další zvláštní omezující podmínky, aby se model úlohy více přiblížil reálnému problému. Čím více je úloha rozšířena do dalších dimensí, tím více možností omezujících podmínek se nabízí. Počet a vzájemná rozmanitost omezujících podmínek však značně komplikuje řešení.

Nejčastějším omezením bývá faktor hmotnosti, jak na straně objektů (jejich hmotnost), tak na straně kontejneru (nosnost). Hmotnost se však v praxi projevuje i v mnohých dalších omezeních – hmotnost na běžný metr, rozložení hmotnosti, poměr zatížení kol a náprav. Dodržení podmínek rozložení hmotnosti je v praxi velmi

[1] Byť by se zdálo, že v praxi je podmínka ortogonálnosti nepoužitelná, lze nalézt její uplatnění mimo jiné v příbuzném řezném problému. Například při výrobě nábytku musí být při řezání desek zohledněn směr dýhování.

důležité, protože nevhodně naložený náklad může způsobit například naklonění lodi nebo nestabilitu silničního dopravního prostředku.

Další častou množinou omezujících podmínek jsou konkrétnější požadavky na umístění a stohování objektů. Například ve skladech a dopravních prostředcích musí zboží ležet na podlaze, nesmí být stohováno, nesmí se na něj stohovat další zboží, nesmí se stohovat těžší na lehčí zboží atd.

Řešení úlohy komplikuje možnost manipulace s objekty, kdy je přípustné naložit objekty v jiné orientaci, než byly zadány. Objekty pak lze umísťovat i v poloze otočené (otáčení kolem vertikální osy) nebo překlopené (otáčení kolem horizontální osy).

Svůj význam může sehrát i priorita objektů či skupin objektů, nebo-li pořadí, ve kterém budou nakládány či vykládány. Zde musí být zaručen volný přístup k postupně vykládaným objektům.

Rozdělení objektů do skupin, které pak mají další vlastnosti může být i z jiných důvodů, například některé druhy nebezpečného zboží nemohou být přepravovány ve vzájemné blízkosti či ve společném prostoru.

V neposlední řadě (zvláště u on-line úloh) může být omezující i čas potřebný pro výpočet řešení.

Je zřejmé, že množství a druhy omezujících podmínek jsou velmi obsáhlé, ne-li nekonečné. Klasický BP problém žádná takováto omezení neuvažuje. S každou omezující podmínkou totiž složitost řešení dále roste.

1.2 Praktické aplikace BP

Pro důkaz, jak může být Bin Packing problém různorodý a v kolika různých oblastech lidské činnosti se vyskytuje, následuje výpis praktických použití. Seznam obsahuje oblasti od těch naprosto zřejmých, až po úlohy, kde by se ani tento problém na první pohled nehledal. Do výpisu jsou zahrnuty i použití příbuzných specializovanějších problémů[2] jako je již zmíněný řezný problém a úloha batohu.

- doprava, skladování a logistika,
- řízení výroby (například automobilový průmysl),
- informatika (rozdělování práce mezi více procesorů nebo serverů, optimalizace umístění dat na paměťové medium atd.),
- časové rozvrhy (například televizní programy atd.),
- architektura,
- chemie a molekulární biologie,
- vojenství,

a mnoho dalších.

[2] o příbuzný problémech je více v kapitole 1.7

1.3 Metody řešení

Řešení, zvláště jednoduchých Bin Packing úloh, je již poměrně dobře propracované. Exaktní metody jsou však použitelné jen pro úlohy o malém rozsahu, jak jsem uvedl výše. V praktických řešeních se proto používají heuristické algoritmy. Pro klasický BP existuje několik jednoduchých a rychlých algoritmů. V poslední době se však zájem řešitelů obrací k metaheuristikám, zejména pak k evolučním algoritmům jako genetické programování a metody mravenčí kolonie. Ty nejúspěšnější metaheuristické algoritmy se zakládají na kombinaci genetických algoritmů s Local Search (5).

To vše se týká většinou jen teoretické úrovně řešení. V praktických úlohách tyto algoritmy často selhávají. Slouží však jako dobrý odrazový můstek a zdroj inspirace pro další specializovanější řešení.

Jako v jiných úlohách, kde se hledá alespoň přibližné řešení, existuje celá řada přístupů. Pro řešení úlohy Bin Packing se dají rozdělit na elementární algoritmy, které se pak používají jako součásti dalších algoritmů, kterými mohou být již zmíněné evoluční algoritmy. Obecně lze použít také gradientní metody, Local Search a další metody prohledávající stavový prostor možných a přípustných řešení, ale existují i některé specializované algoritmy, na které bude zaměřena bližší pozornost.

1.3.1 Heuristické algoritmy pro řešení klasického BP problému

Snad nejjednodušším algoritmem pro řešení klasického BP problému, tedy BP v jedné dimensi, je algoritmus „Next fit“ (NF). Jedná se o naprosto zřejmý postup, ale poprvé pod tímto názvem ho zveřejnil pravděpodobně D. S. Johnson (7) (10) (18). Jedná se o ohraničený on-line algoritmus, kde se, zjednodušeně řečeno, objekty umísťují vždy do jednoho „právě otevřeného“ kontejneru. Pokud se tam již další objekt nevejde, kontejner se uzavře a odloží a objekty se umísťují do dalšího kontejneru. Tento algoritmus má tedy lineární složitost $O(n)$. Jeho nevýhoda spočívá právě v tom, že pracuje vždy pouze s jedním kontejnerem a je poměrně neefektivní.

Tuto nevýhodu odstraňuje další algoritmus, nazývaný “First fit” (FF). Ten umožňuje umístit objekt do kteréhokoliv doposud “otevřeného” kontejneru. Postup je takový, že procházíme kontejnery již použité v řešení a objekt umístíme do prvního, do kterého se objekt vejde. Pokud žádný takový kontejner není, bude založen nový a objekt bude umístěn do něho. Na první pohled algoritmus pracuje s kvadratickou složitostí, ale při použití vhodné struktury dat je možné dosáhnout složitosti $O(n \cdot \log n)$. Tato složitost odpovídá lineární složitosti při vkládání objektů a složitosti nejlepších třídících algoritmů (např. Quick Sort) použitých na uspořádání kontejnerů.

Analogicky jako „First fit“ pracuje algoritmus „Last fit“ (LF), který stejně tak prochází seznam kontejnerů, ale objekt vloží až do toho posledního, do kterého se vejde. Stejná je i minimální dosažitelná složitost algoritmu - $O(n \cdot \log n)$.

Podobným principem pracují i další algoritmy. Za velmi nadějný byl považován algoritmus „Best Fit“ (BF). Algoritmus umístí objekt do toho nejzaplněnějšího kontejneru. Podrobné testy a analýzy však ukazují (2), že BF algoritmus prokazatelně lepších výsledků než FF algoritmus nedosahuje. Výpočetní složitost je opět $O(n \cdot \log n)$.

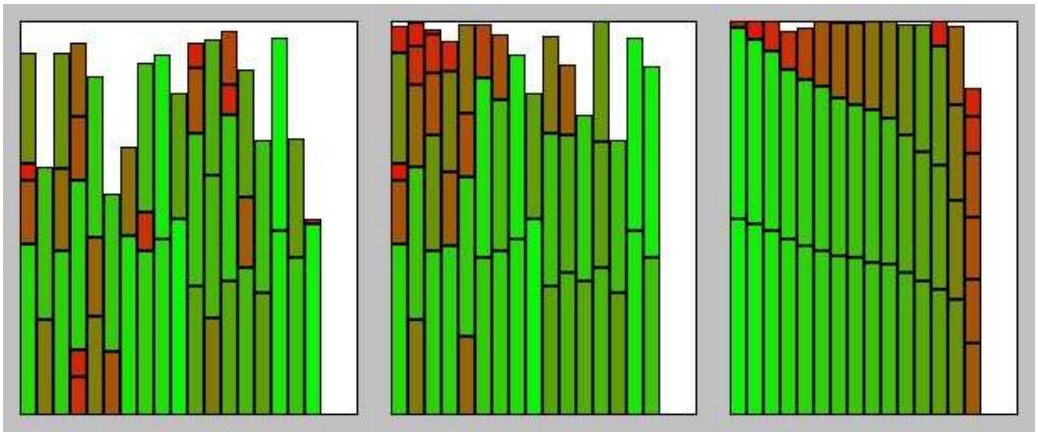
Podobné jsou algoritmy „Worst fit“ (WF) a „Almost Worst Fit“ (AWF). Ty neumísťují objekt do toho nejvíce plného kontejneru jako BF, ale naopak do kontejneru nejprázdnějšího, respektive druhého nejprázdnějšího kontejneru v případě AWF.

Tyto jednoduché heuristické metody mají i tu výhodu, že jsou schopny pracovat on-line. Nevýhodou je, že nedosahují příliš dobrých výsledků. J. Ullman dokázal, že výsledky tohoto algoritmu se mohou lišit od optimálního řešení až o 70% (6). Poměrně velké kvalitativní zlepšení nastává až po úpravě pořadí vstupujících objektů, tedy určitém setřídění. Tímto se však ztrácí výhoda on-line přístupu.

Pro modelování, analýzu a porovnání algoritmů je obecně možno objekty uspořádat těmito způsoby:

- neuspořádaná, náhodně uspořádaná data (odpovídá on-line vstupům),
- pravidelně uspořádaná data – vstupní objekty jsou pravidelně uspořádány tak, aby mezi hodnotami sledovaných parametrů objektů vznikly co možná největší vzájemné rozdíly,
- vzestupné uspořádání vstupů – podle hodnot sledovaného parametru objektů od nejmenších hodnot po největší,
- sestupné uspořádání – od největších hodnot po nejmenší. Do algoritmu potom vstupují nejdříve objekty s maximální hodnotou sledovaného parametru.

Po použití třídících algoritmů pro vhodné uspořádání vstupujících objektů je výsledek algoritmů již znatelně lepší než při neutříděných vstupech. V roce 1973 D. Johnson ukázal, že výsledky nad utříděnými daty se od optimálního řešení neliší více než o 22% (7). Kromě toho také ukázal, že tyto jednoduché algoritmy bohužel lepších výsledků ani nedosáhnou. To potvrdila i další práce „Approximation algorithms for Bin Packing“ pánů Coffmana, Gareye a Johnsna (2), kde jsou tyto algoritmy analyzovány z hlediska jejich výkonnosti v těch nejhorších možných případech (worst case analysis).



Obrázek 1.1 Porovnání elementárních algoritmů (10)

Na obrázku Obrázek 1.1 je pomocí výstupů z počítačového programu „Bin Packing“ (10) ukázána práce těchto základních algoritmů. Množina objektů byla náhodně vygenerována. Barva objektů odlišuje jejich rozměr. Vlevo jsou výsledky dosažené algoritmem Next Fit nad neutříděným seznamem objektů. Zde bylo použito 17 kontejnerů (sloupečků). Uprostřed jsou výsledky algoritmu First Fit, opět nad neutříděným seznamem objektů, - 16 použitých kontejnerů. Vpravo jsou výsledky získané opět algoritmem First Fit, ale tentokráte nad seznamem objektů setříděným sestupně podle velikosti. Zde bylo zapotřebí pouhých 15 kontejnerů. Využití jejich kapacity je zřejmé právě z obrázku Obrázek 1.1.

1.3.2 Martello-Tothův algoritmus

Kromě těchto jednoduchých heuristických algoritmů lze Bin Packing problém řešit pomocí dalších specializovaných algoritmů. Jedním z nich je i méně známý Martello-Tothův algoritmus (MTA)(13). Tento algoritmus je oproti výše uvedeným pomalejší a z hlediska výpočetní techniky paměťově náročnější, což je nevýhodné zvláště u úloh většího rozsahu, ale zato podává lepší výsledky.

Základem MTA je představa, že určité varianty řešení dominují jiným. Pokud jedno řešení dominuje, zjednodušeně to znamená, že obsahuje více objektů než ostatní varianty. MTA se pak snaží iterativně nalézt takové varianty řešení (tedy takové uspořádání objektů v kontejnerech), které dominují všem ostatním. V každé iteraci je část řešení prohlášena za definitivní, a tím se rozměr úlohy postupně zmenšuje. Takto podaný algoritmus pracuje s exponenciální výpočetní složitostí, což není pro praktické řešení rozsáhlejších úloh příliš vhodné. Této komplikaci se lze vyhnout modifikací algoritmu, například zavedením dalších předpokladů či úpravou (zjednodušení) pravidel pro stanovení dominujícího kontejneru. Tím ovšem dochází ke vzdalování se od optimálního řešení.

1.3.3 Local Search

V roce 1999 Faroe, Pisinger and Zachariasen použili na řešení BP přístup založený na Local Search (5). Tento, v podstatě greedy algoritmus, umísťuje objekty do kontejneru pouze podle rozměrových kritérií. První řešení je uloženo v paměti a algoritmus pracuje lépe s každým dalším přidaným objektem s použitím předchozího řešení. Část tohoto řešení pak může být změněna.

Nevýhodou algoritmu může být, že se nalezne pouze lokální minimum účelové funkce a algoritmus není dále schopen vyprostit se z této oblasti stavového prostoru řešení. Tento postup je proto nutno kombinovat ještě s dalšími, například použít metodu tak zvaného „simulovaného žíhání“.

1.4 Hledání dalších odvozených řešení

Výše popsané algoritmy představují jen část různých postupů a přístupů k řešení tohoto problému. Nejsou vždy schopny vygenerovat řešení, které kvalitativně odpovídá reálným požadavkům. Tyto algoritmy však mohou také sloužit jen pro získání nějakého výchozího přípustného řešení. Na řadu potom přicházejí další metody a algoritmy, které upravují, zkvalitňují či případně opravují toto výchozí řešení.

1.4.1 Výměnné heuristiky

Metody této skupiny upravují a vylepšují již existující přípustné řešení, které je získané některým výše popsaným heuristickým algoritmem. Jedná se opět o heuristické algoritmy, takže ani jejich aplikováním není zaručena optimálnost získaného řešení. Tyto metody jsou založeny na vyměňování částí řešení při zachování její přípustnosti, přičemž je kontrolováno, zda touto výměnou nedošlo ke zkvalitnění řešení.

Konkrétně pro Bin Packing to znamená, že se pokouší vyměňovat objekty mezi kontejnery v rámci jednoho řešení nebo mezi různými variantami řešení. Aplikace samotných těchto metod je poměrně jednoduchá, ale bývá problém vytvořit mechanismy pro rychlou a snadnou kontrolu zachování přípustnosti řešení.

1.4.2 Genetické algoritmy

Genetické programování a genetické algoritmy se svým přístupem významně liší od běžných algoritmů. Používají množinu řešení (populace), která se v dalších iteracích (generacích) mezi sebou kříží, mutuje, vybírá nejlepší pro další generace atd. Každé řešení se zde nazývá chromozom a jeho kvalita se určuje podle funkce zvané fitness (20).

Existuje mnoho různých přístupů a postupů práce s těmito chromozomy. Použití genetických algoritmů je obecné, ale pro každý typ úlohy je nutné zvolit či sestavit vhodná pravidla pro operace selekce, mutace a křížení. Je nutné vhodně určit podobu chromozómu a sestavit vhodnou funkci fitness.

Pro řešení problému BP poprvé aplikovali genetické programování v 1992 A. Corcoran a R. Wainwright (3). Použili tenkrát jediný neomezený kontejner (nekonečná výška, tedy i objem). Objekty byly před tvorbou první generace uspořádány náhodně. První řešení byla generována pomocí algoritmů „Next Fit“ a „First Fit“. Dosažená výška v kontejneru určovala kvalitu řešení - čím byla nižší, tím představovala lepší řešení. Podle tohoto jednoduchého kritéria byla nekvalitní řešení odložena a ta kvalitnější se kombinovala do dalších generací. Výpočetní složitost a náročnost na čas výpočtu ve svém výzkumu neporovnávali.

1.4.3 Ant colony

Ant Colony Optimization (ACO) se inspiruje v biologii podobně jako genetické algoritmy. Používá umělé modelované mravence, kteří náhodně objevují nová řešení na základě heuristických informací o umělé „feromonové stopě“. Feromonová stopa je umocňována počtem průchodů mravenců, tedy kvalitnějším řešením. Tento přístup použil poprvé M. Dorigo pro řešení úlohy obchodního cestujícího.

Ant Colony je populární evoluční technika pro hledání řešení kombinatorických úloh. V případě Bin Packingu je cílem určit, který objekt bude přiřazen do kterého kontejneru. To se určuje pomocí feromonové matice, která v podstatě vyjadřuje „oblíbenost“ tohoto přiřazení. Feromonová stopa vzniká průchodem mravence, tedy použitím řešení. Silná feromonová stopa je mezi mravenci oblíbenější. Každým dalším použitím se feromonová stopa zesiluje, po každé iteraci se však také „odpařuje“.

Podle výzkumů Levina a Ducatella (12) podává kombinace této techniky s lokálním prohledáváním stavového prostoru řešení (Local Search) poměrně dobré výsledky

1.4.4 Kombinace metod

Při řešení praktických úloh by bylo vhodné kombinovat tyto různé algoritmy. Nějaká obecná koncepce a kritéria pro výběr, který algoritmus ve kterém případě použít však neexistuje[3]. U heuristických algoritmů je toto velmi obtížné, protože výsledky, které podávají, často velmi závisí na charakteristikách vstupujících objektů a parametrech použitých při výpočtech. Tyto parametry lze chápat například jako váhové koeficienty jednotlivých částí účelové funkce či celou řadu dalších nastavení.

1.5 Neuronové sítě

V posledních letech prudce vzrostl zájem o obor výpočtů pomocí neuronových sítí. Tento zájem je podpořen požadavky výpočtů algoritmů, které klasická výpočetní technika není schopna řešit. U umělých neuronových sítí (UNS) je vzorem chování odpovídajících biologických struktur, je tedy snaha napodobit strukturu a činnost biologického mozku, protože dokáže:

- zjednodušovat,

[3] Kromě klasického kombinování gradientních algoritmů s principy simulovaného žíhání atp.

- zobecňovat,
- používat „obecné“ pro konkrétní úlohy.

Přesný algoritmus práce přírodních neuronových systémů není doposud znám, přesto experimentální výsledky na modelech těchto systémů již dávají poměrně slibné výsledky.

1.5.1 Význam umělých neuronových sítí

Neuronová síť je jedním z výpočetních modelů používaných v umělé inteligenci. Jedná se o strukturu určenou pro distribuované paralelní zpracování informací, která se skládá z vysokého počtu samostatných výkonných prvků (20). Z osvědčených praktických využití lze jmenovat především rozpoznávání psaného písma, rozpoznávaní lidských tváří, či řízení vozidla.

Technologie neuronových sítí však rozhodně není všelékem k řešení všech problémů. Jejich výhoda se projevuje zejména v případech řešení úloh s nepřesnými algoritmy, nebo úloh, kde není kompletní sada algoritmů pro řešení nebo kde jsou algoritmy příliš složité pro matematické formulace problému – řešení je přibližné, nepřesné a zjednodušené. Naopak nevhodné uplatnění umělých neuronových sítí je v některých konkrétních exaktních výpočtech.

Hlavní trendy rozvoje umělých neuronových sítí jsou v oblastech:

- modelování funkcí „informačních systémů“ živých organismů,
- výzkumů procesů učení a adaptivity, jejich testování,
- optimalizace topologie systémů,
- predikce časových řad (v energetice, finančnictví, vojenství, meteorologii, atd.),
- analýza obrazů a zvuků, analýza vícerozměrných a složitých signálů,
- komprese a kódování dat,
- adaptivní řízení složitých systémů,
- systémy pro rozhodování.

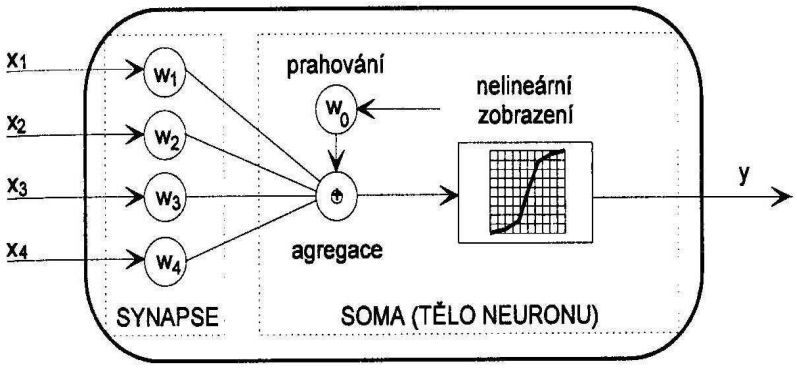
Při výzkumu neuronových sítí se uplatňují dva přístupy:

- Analytický, který zkoumá reálný svět a nachází modely funkcí a metody jejich simulace.
- Syntetický, který navrhuje struktury UNS schopné plnit a realizovat požadované funkce.

1.5.2 Model umělého neuronu

Funkcí biologického neuronu je shromáždit elementární informace ze vstupů (čili z výstupů ostatních s ním spojených neuronů), zpracovat je a prostřednictvím svého výstupu je postoupit dále.

Analogicky se UNS se skládají z umělých („formálních“) neuronů, které jsou vzájemně propojeny a navzájem si předávají „signály“ a transformují je pomocí určitých přenosových funkcí. Neuron je tedy elementární jednotka pro zpracování informace, která má libovolný počet vstupů, ale pouze jeden výstup.



Obrázek 1.2 Funkční schéma neuronu (15)

xi	vstupy neuronu
wi	synaptické váhy - uložení zkušeností do neuronu
w0	prahová hodnota aktivace neuronu
f(x)	přenosová funkce neuronu (někdy aktivační funkce)
y	výstup neuronu

Konkrétně tento McCullochův a Pittsův model neuronu na obrázku Obrázek 1.2 Funkční schéma neuronu (15) byl prvním formálním modelem neuronu, který byl použit pro modelování jak biologického, tak i umělého neuronu. Byl nazván binárním prahovým neuronem. Tento model má pevný počet vstupů a neměnnou prahovou hodnotu. Do neuronu přichází vzruchy (vstupy) xi představují binární hodnoty (0 pokud nepřichází vzruch, 1 pokud přichází vzruch). Velikost vah wi vyjadřuje zkušenosti zaznamenané v neuronu, tedy takovou „lokální paměť“. Čím je tato hodnota vyšší, tím je daný vstup důležitější. Působení vzruchů se v neuronu sčítá. Pokud součet dosáhne alespoň prahové hodnoty w0, neuron na výstupu reaguje hodnotou 1, jinak hodnotou 0. Prahová hodnota w0 tedy řídí celkovou aktivaci neuronu. To lze popsat vztahem:

$$y = f\left(\sum_{i=1}^{N}\left(w_i \cdot x_i\right) + w_0\right) \qquad 1.1$$

kde, konkrétně pro tento model je funkce f je odvozena od matematické funkce signum, která nabývá hodnoty +1 pro hodnoty větší nebo rovno nule a –1 pro hodnoty menší než nula.

Binární prahový neuron dokáže provádět jenom jednoduché logické funkce. Pokud bude z těchto neuronů sestavena vhodná síť, může řešit i složitější logické operace.

Jak tomu bývá, není vše tak ideální. Neexistuje totiž obecný algoritmus pro určení velikosti a topologie takové sítě ani pro stanovení prahových hodnot zúčastněných neuronů. Neurony poskytují skokové binární výstupy (tedy diskrétní hodnoty) a ne výstupy spojité, jak vyžaduje většina reálných aplikací. Z těchto důvodů se tento model neuronu nedal v mnoha případech použít. To vedlo ke hledání vhodnějších a obecnějších modelů.

Obecně mají na funkci neuronu vliv:

- napojení neuronu, tedy topologie neuronové sítě,
- přenosové vlastnosti, tedy vliv vah vstupů a prahování signálu,
- druh přenosové funkce *f*,
- funkční vlastnosti – pamatování, učení.

Obecně lze podle povahy vstupních a výstupních dat neurony dělit na:

- binární,
- spojité.

Podle typu neuronu a typu neuronové sítě se použije vhodná přenosová funkce *f*. Ta může být například:

- skoková (viz popsaný model),
- sigmoidální (logistická),
- hyperbolické tangenty,
- radiální báze.

1.5.3 Topologie

Topologie neuronové sítě je určena jejím grafem, jehož jednotlivé uzly odpovídají výkonným prvkům (neuronům) a hrany grafu propojením mezi nimi.

Podle způsobů propojení neuronů existuje více různých architektur UNS, například:

- Perceptron,
- vrstevnatá neuronová síť,
- rekurentní neuronová síť,
- Kohonenovy mapy,
- modulární neuronové sítě.

Vzhledem k složitosti geometrické struktury sítě, lze některé výkonné prvky sdružovat podle shodných vlastností do samostatných oddílů, které pak tvoří základní neuronovou síť. Tyto oddíly mohou být:

- vstupní vrstva – pouze přenáší vstupy dovnitř sítě, nemají svou lokální paměť,
- vnitřní (skryté) vrstvy – zpracovávají vstupy podle určených přenosových, mají své lokální paměti, implementují procesy adaptibility a učení,
- synchronizační obvody, které synchronizují prvky jednoho oddílu,
- výstupní vrstva - předává jednotlivé složky výstupů ven z neuronové sítě.

1.5.4 Učení

Neuronová síť je popsána maticí resp. soustavou váhových koeficientů. Tyto váhy w_i jsou časově proměnné. Učení neuronových sítí tedy spočívá v nastavení vah spojení w_i tak, aby sít vytvářela požadované výstupní data na zadaná data vstupní.

Učení se rozlišuje na:

- asociativní – kde je cílem extrakce vztahů mezi vstupními daty, resp. mezi vstupy a výstupy,
- neasociativní – kde je předkládán jistý jednoduchý „datový stimul“ a účelem je si později tyto stimuly vybavit.

Existují dva základní způsoby učení:

- učení s učitelem – uživatel iterativní procesem porovnává aktuální výstup z neuronové sítě s požadovaným výstupem a koriguje váhy tak, aby se tento rozdíl minimalizoval,
- učení bez učitele – váhy se nastavují tak, aby byly výstupy sítě konzistentní, tedy aby síť poskytovala adekvátně stejné výstupy při podobných vstupech; musí být implementován algoritmus automatického učení.

Někde mezi těmito základními způsoby učení je hodnocené učení, kdy trénovaná neuronová síť nedostává pro své učení k dispozici žádné učební vzory, ale dílčí výsledky procesu učení jsou průběžně hodnoceny takovým způsobem, že proces učení vede k žádoucímu cíli.

Problémem učení může být přetrénovanost, kdy síť už neurčuje charakteristické prvky pro výsledky, ale každý vstup trénovacích dat se může stát samostatným výstupem.

1.5.5 Interpretace znalostí

Použití neuronových sítí s sebou nese problém interpretace jejich znalostí. Znalosti získané neuronovými sítěmi jsou podchyceny její topologií a váhami vazeb mezi neurony, takže pro uživatele jsou zcela nesrozumitelné. S tím souvisí i neschopnost neuronových sítí podávat vysvětlení získaného řešení. Z neuronové sítě se tedy stává černá skříňka, do které není vidět.

1.6 Datové struktury

Pro matematické, tím spíše počítačové zpracování je nutno mít jednotlivá řešení nějakým způsobem zaznamenána a podchycena ve vhodných datových strukturách. Použité datové struktury mají významný vliv na přehlednost, implementaci algoritmů, zábor paměti a rychlost výpočtů. Z těchto důvodů je potřeba zaměřit se i na tuto oblast problému.

Pro počítačové zpracování se nabízí použití objektově orientovaného programování. Za účelem provázání objektů je nutné vytvořit přehlednou a dostatečně rychlou referenční strukturu. Naštěstí je možné nechat se inspirovat oborem počítačové grafiky, kde je datový záznam objektů a jejich vzájemné uspořádání již dlouho a poměrně úspěšně řešen (22).

1.6.1 Rozhodovací stromy

Rozhodovací stromy obecně jsou analytické nástroje sloužící k nalezení pravidel a vztahů v datové struktuře pomocí systematického rozdělování a větvení na nižší úrovně. Pracuje se tedy principem „rozděl a panuj“. Strom se skládá z uzlů. Uzel na nejvyšší úrovni je označován pojmem kořenový. Vnitřní uzly přestavují proces dělení výchozí množiny podle určitého atributu, jehož výsledkem je další větvení. Listové uzly reprezentují finální rozdělení objektů na jednotlivé třídy. Cílem rozhodovacích stromů je tedy identifikovat objekty, popsané různými atributy, do tříd. (4)

Rozhodovací stromy jsou vhodné pro úlohy, ve kterých má být provedena klasifikace nebo předpověď. Užitečné jsou v oblastech, ve kterých můžeme hodnoty proměnných rozdělit do relativně malého počtu skupin.

1.7 Příbuzné a související úlohy

1.7.1 Úloha batohu

Problém batohu je jedním z nejznámějších NP-těžkých problémů. V literatuře lze najít velké množství rozličných variant, které mají obecně různé nároky na řešení i jeho algoritmus. Nejznámější varianta je optimalizační. Cílem úlohy je určit (pod)množinu objektů *X* a naložit je do batohu, aby platilo:

$$\sum_{i=1}^{n} x_i \cdot v_i \leq M \qquad 1.2$$

což znamená, že batoh nebude přetížen. Dále se požaduje, aby výraz

$$\max \rightarrow \sum_{i=1}^{n} x_i \cdot c_i \qquad 1.3$$

n	počet objektů
M	kapacita batohu
v	hmotnosti objektů
c	ceny objektů
x	binární proměnná značící, zda je objekt v batohu

nabýval maximální hodnoty, tedy aby celková cena objektů v batohu byla maximální.

Dále existuje například exaktní problém batohu, který vyžaduje, aby se součet vah objektů rovnal přesně kapacitě batohu. Inverzní problém batohu naopak vyžaduje, aby byl součet vah minimální při zachování předem určené celkové ceny naložených objektů (20).

Úloha batohu tedy na rozdíl od úloh BP neřeší vzájemné uspořádání objektů v kontejneru.

1.7.2 Řezný problém

Řezný problém, nebo také úloha o dělení materiálu řeší dělení nějakého materiálu na předem definované části tak, aby odpad po řezání byl minimální. Jedná se tedy o problém k problému Bin Packingu v jistém smyslu inverzní.

Formulace standardního řezného problému předpokládá výchozí seznam objednávek, kde každá vyžaduje stanovený počet kusů nějakého materiálu daných rozměry. Cílem je sestavit seznam všech možných kombinací (často se říká "vzory") kusů, a počty, kolikrát má být každý vzor použit. Lineární model úlohy je pak:

$$\min \rightarrow \sum_{i=1}^{n} c_i \cdot x_i \qquad 1.4$$

s dodržením omezujících podmínek:

$$\sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \cdot x_i \geq q_j \qquad 1.5$$

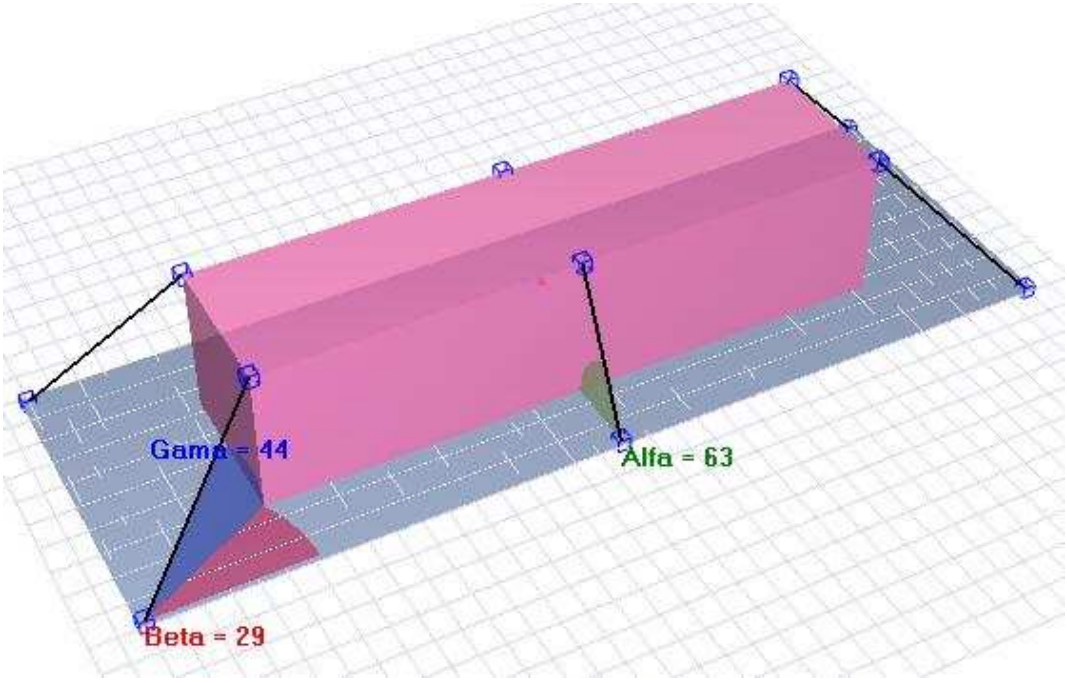
m	počet objednávek
q	počet kusů požadovaných v objednávce
n	počet kombinací (vzorů)
x	počet výskytů vzoru
c	náklady (ztráty) z konstrukce vzoru
a	počet, kolikrát se daný vzor i vyskytne u dané objednávky j

Řezný problém se může vyskytovat, stejně jako BP problém, v dalších dimensích, jako 2D a 3D. (20)

1.7.3 Stabilita a vliv dynamiky na zásilku

Při jízdě dopravního prostředku působí na naložený náklad různé dynamické vlivy. Jedná se především tíhovou sílu a o setrvačné síly způsobené zrychlením nebo zpomalením či dostředivým zrychlením při průjezdu zatáčkou. Proti působení těchto sil je potřeba náklad vhodně zabezpečit (zajistit). Pro správné dimenzování tohoto zabezpečení je třeba působící síly matematicky zjistit.

Německá směrnice pro zajišťování nákladu na dopravních prostředcích VDI 2702 (19), která je považována za evropský standard v tomto oboru, podrobně popisuje způsob výpočtů i samotné vzorce pro výpočet působících sil, definuje způsoby zajištění nákladu a předkládá odpovídající vzorce pro výpočet sil potřebných k dostatečnému zajištění nákladu. Cílem disertační práce není tuto směrnici opisovat, v tomto místě i v průběhu dalšího řešení však na ni budu odkazovat jakožto na nejkomplexnější materiál zabývající se touto problematikou.



Obrázek 1.3 Fixace naloženého zboží (Software Fixload)

1.8 Praktické softwarové aplikace

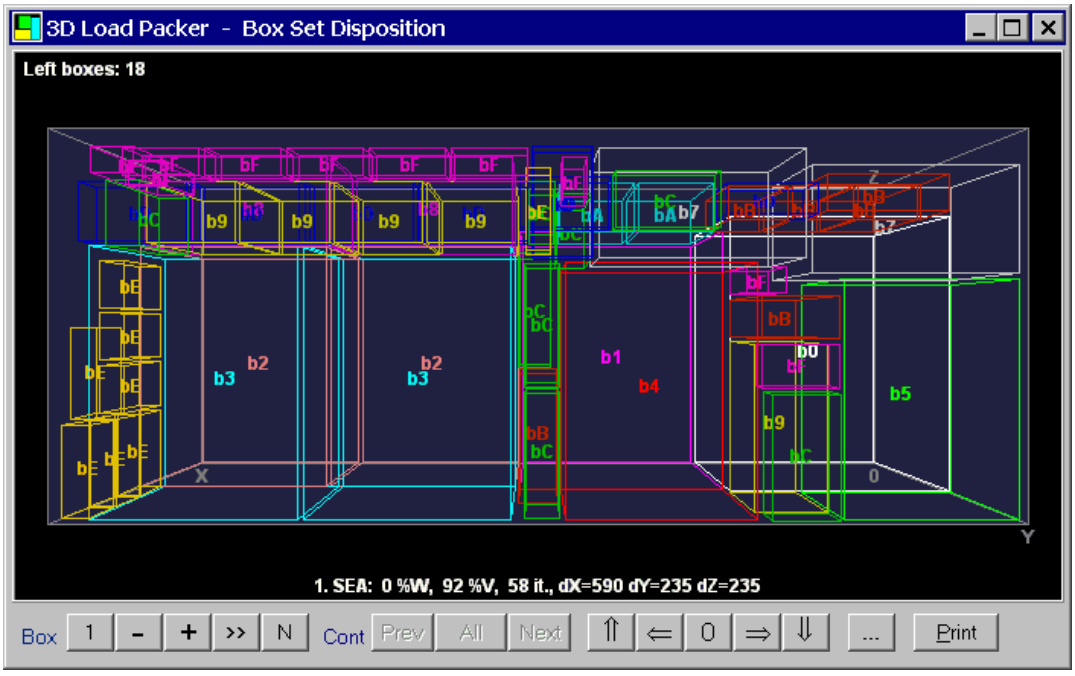
Řešení úloh Bin Packingu je v praxi celkem běžné. Nejčastěji se však jedná o přibližná „manuální“ řešení. Použití automatizovaného řešení pomocí výpočetní techniky vyžaduje značně sofistikovaný přístup a proto není natolik rozšířeno. To je pochopitelné, protože existuje obrovské množství typů tohoto problému. Jen pro úlohu BP, která řeší nakládání zboží na dopravní prostředky si lze představit, že pro téměř každý druh zboží existují normy či doporučení, jakým způsobem ho nakládat a v dopravních prostředcích zajišťovat. Proto je velmi obtížné sestavit obecný matematický model a následný počítačový program pro řešení úlohy Bin Packingu. To je i důvodem, proč vývojáři často vybavují tyto optimalizační programy také

podpůrným interaktivním grafickým rozhraním, pomocí něhož mohou uživatelé snadno „doladit“ umístění zboží v ložném prostoru tak, aby ložný plán splňoval praktické požadavky.

Podle mých průzkumů neexistuje v České republice původní SW, který by tuto problematiku o zadaném rozsahu zpracoval a automaticky generoval alespoň částečná řešení. V zahraničí existuje několik SW produktů, které toto alespoň částečně splňují. Dále uvedu stručné hodnocení některých z nich.

1.8.1 Astrokettle

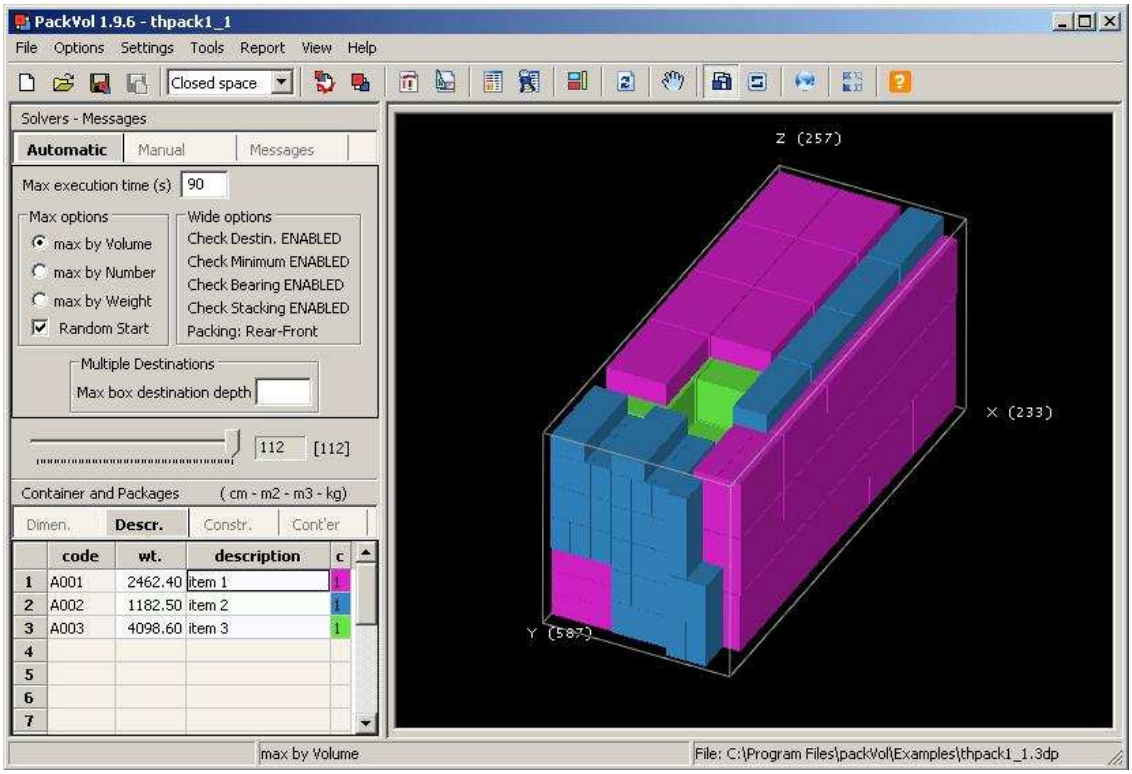
Astrokettle je společnost nabízející produkty nejen pro optimalizaci nakládání. Program 3D Load Packer umožňuje rychlou optimalizaci, při dodržení základních omezujících podmínek. Formální, ale velkou nevýhodou je poněkud nepřehledné uživatelské prostředí.



Obrázek 1.4 3D Load Packer (www.astrokettle.com)

1.8.2 packVol

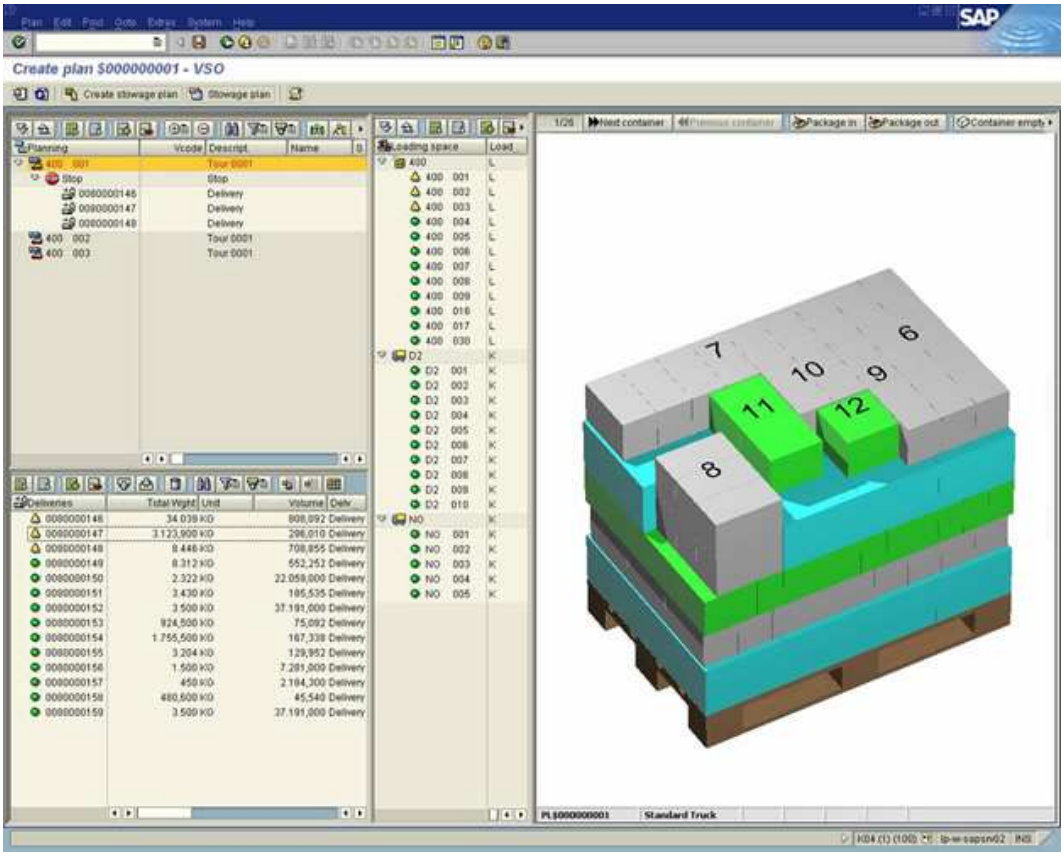
PackVol je jednoduchá aplikace, která umožňuje automatickou optimalizaci i manuální nakládání. Optimalizace je možná podle účelových funkcí maximalizace využití prostoru, maximalizace využití nosnosti nebo maximalizace počtu naložených objektů. Respektuje rozměrové a hmotnostní omezující podmínky. Podporuje snadnou tvorbu ložných plánů, má příjemné uživatelské prostředí.



Obrázek 1.5 packVol (www.packvol.com)

1.8.3 Logiplan

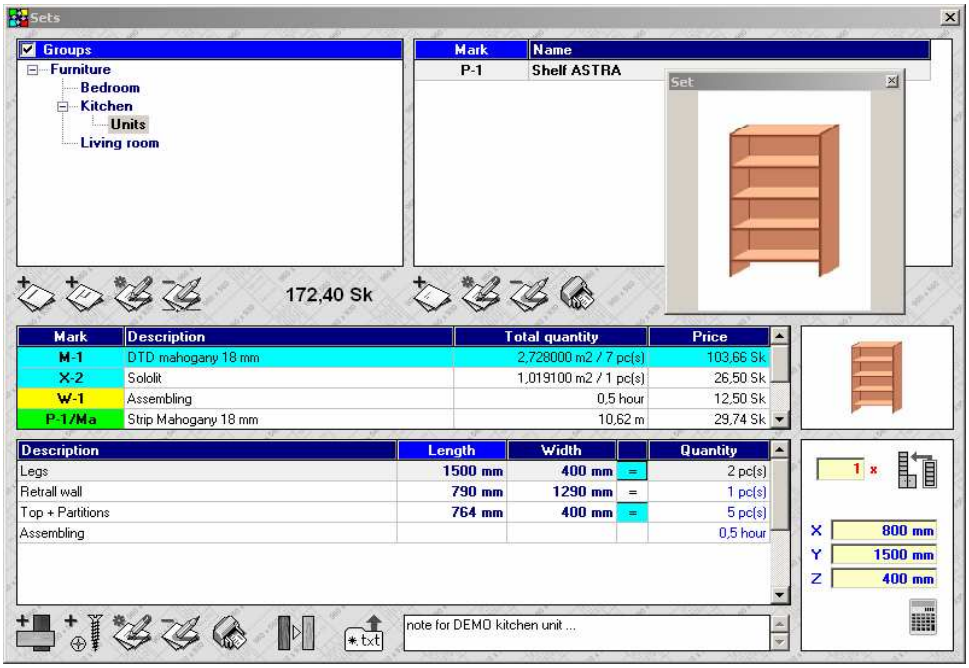
Program Load Designer společnosti Logiplan patří asi ke špičce ve svém oboru. Umožňuje optimalizaci nakládky jak kubických, tak cylindrických materiálů na libovolné typy a počty palet a poté optimalizovat plán nakládky dopravního prostředku či kontejneru. Plán nakládky lze optimalizovat např. podle vykládky, využití místa, typu kontejnerů, nákladů na cestu a spousty dalších. Samozřejmostí je kontrola celé řady omezujících podmínek: maximálního zatížení, výpočet zatížení náprav. Celé prostředí je plně grafické včetně grafického zobrazení nakládky a vykládky, paletizace apod. Součástí programu je i napojení na další systémy, jako jsou např. SAP. To představuje velkou výhodu při použití již jednou zadaných údajů.



Obrázek 1.6 Load Designer (www.logiplan.de)

1.8.4 Optimik

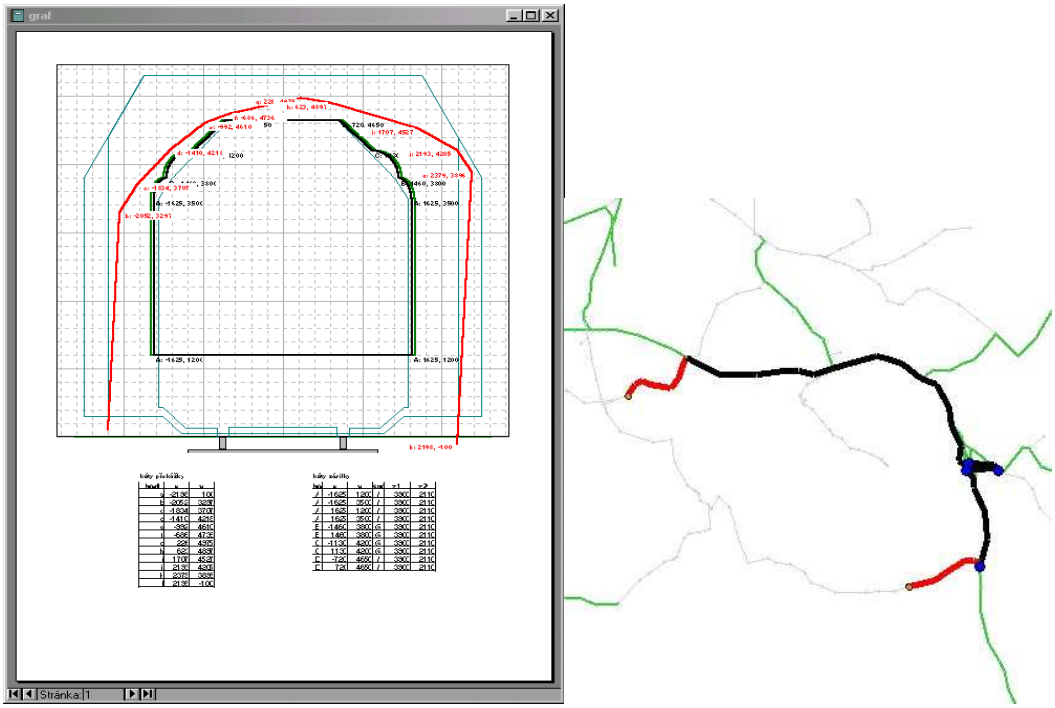
Zajímavý software ze Slovenské republiky, řešící příbuznou skupinu řezných úloh. Program šetří materiál a čas přípravy výroby pomocí návrhu nářezových plánů pro zakázkovou výrobu nábytku, stříhání plechu, řezání skla atd. Optimik umožňuje nastavit orientaci hlavních řezů, rychlou optimalizaci s průměrnou efektivitou až 90%, využití odřezků a celou řadu dalších uživatelských funkcí.



Obrázek 1.7 Optimik (www.rksoft.sk)

1.8.5 Mimoza

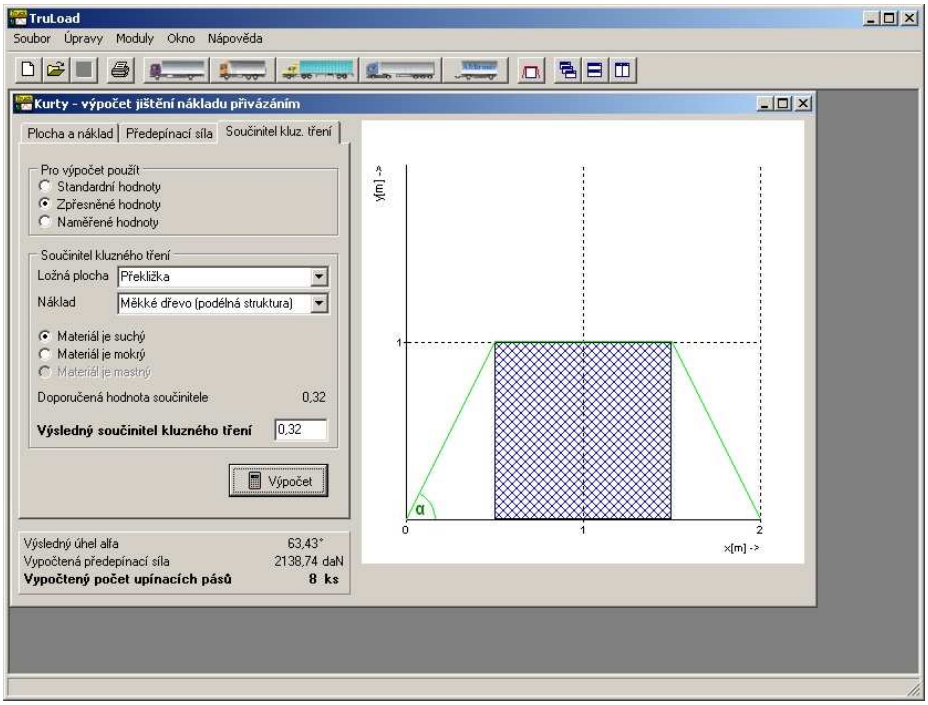
S problémem nakládání souvisí také problematika, která se nazývá překročení ložné míry (v železniční dopravě se používá zkratka PLM). V praxi běžně nastává a řeší se situace, kdy náklad svými rozměry překročí rozměry dopravního prostředku. V případě ČD Cargo se prostorovou průchodností zásilky zabývá informační systém Mimoza (Mimořádné zásilky).



Obrázek 1.8 Mimoza (Oltis Group a.s.)

1.8.6 Excolo - TruckLoad

Truckload je program zaměřený na problematiku zajištění nákladu na silničních nákladních vozidlech, výpočtem stanoví potřebné množství upínacích pásů, je-li náklad zajištěn přivázáním. Základním předpokladem přesného výpočtu je dosazení odpovídajících přepravně technických parametrů vozidla, nákladu, předepínací síly vázacích prostředků a součinitele kluzného tření.



Obrázek 1.9 TruLoad (Excolo)

1.8.7 Shrnutí analýzy SW

Po prozkoumání a otestování těchto a některých dalších aplikací jsem došel k závěru, že se často jedná pouze o poměrně jednoduchou optimalizaci, kdy algoritmus určí jakým způsobem naložit zboží do zadaného dopravního prostředku. Někdy chybí možnost zadání praktických omezujících podmínek či možnost uživatelského zásahu do řešení. Proto je praktické použití většiny z těchto softwarových nástrojů diskutabilní.

Z praktického hlediska je potřebné, aby takovéto SW produkty obsahovaly kromě prosté optimalizace využití ložného prostoru také nástroje pro kontrolu zajištění zboží na dopravních prostředcích a modely působení dynamických vlivů na zásilku při přepravě. Velmi výhodné by bylo, kdyby se současně při optimalizaci umístění zboží kalkulovalo také se zajištěním jeho stability a optimalizací použití fixačních prostředků vzhledem k jejich umístění i ceně. Nutností je interaktivní grafické rozhraní.

2 Cíle disertační práce

Z analýzy současného stavu vědeckého poznání problematiky Bin Packingu vyplývá, že díky vysoké výpočetní složitosti neexistuje postup, kterým lze získat exaktní řešení rozsáhlých úloh v reálném čase. Proto byla vyvinuta celá řada heuristických algoritmů, které jsou buď obecněji použitelné, ale pouze za cenu nepřesných výsledků, nebo se úzce specializují na řešení specielních typů úloh.

Cílem disertační práce je proto navrhnout nový obecný postup, pomocí kterého bude možné získat kvalitní řešení při daném stupni obecnosti použití. Na vytyčeném modelu úlohy budou navrženy postupy, metody a algoritmy, kterými by bylo možné dosáhnout kvalitnějších řešení. Kvalitu řešení lze přitom chápat ze dvou hledisek:

- přiblížení se optimu při řešení praktických úloh nakládání,
- zrychlení výpočtu těchto úloh.

V práci bude uvažován podrobný model úlohy. V něm bude kladen důraz zejména na z praxe vyplývající omezující podmínky při zachování jeho obecné flexibility.

Stěžejní částí vlastního řešení bude příprava metod pro nalezení empirických vztahů a pravidel mezi výsledky analýzy vstupujících objektů do úlohy a parametry používanými při výpočtu. Na základě těchto pravidel bude možné automaticky přizpůsobit parametry optimalizačních metod charakteru nakládaných objektů a tím pro daný typ úlohy dosáhnout kvalitnějšího řešení. To ovšem bude vyžadovat vytvoření specielních algoritmů pro optimalizaci nakládky.

Navržené metody a algoritmy budou implementovány v rámci vytvořených softwarových aplikací. To umožní jejich precizní otestování a některé názorné grafické výstupy. Tyto softwarové aplikace budou rovněž součástí disertační práce.

Disertační práce tedy rozšíří metodický aparát pro řešení úloh Bin Packingu nejen o obecný metaalgoritmus, ale i o nové dílčí metody a algoritmy. Současně však také nabídne možnosti praktického využití, především v dopravě a logistice.

3 Model vlastního řešení

Před zahájením vlastního řešení je třeba určit výchozí předpoklady a omezující podmínky neboli definovat konkrétního typ BP problému. V analytické části práce bylo uvedeno členění úloh Bin packing. V tomto duchu bude specifikován typ úlohy, kterým se budu nadále v této disertační práci zabývat.

Dalším úkolem při sestavě vlastního modelu úlohy je charakteristika požadovaných výstupů, tedy co bude řešením. Při řešení budu vycházet z hlavních praktických požadavků a podmínek. Budu tedy řešit problém, jak co nejvýhodněji naložit objekty do předem určené množiny kontejnerů, které jsou k dispozici. Cílem je, aby přípustně naložená množina objektů byla přiřazena do kontejneru optimálně z hlediska definované účelové funkce, a to při dodržení daných omezujících podmínek.

V rámci popisu modelu jsou průběžně podávány informace o tom, která data budou pro řešení úlohy potřebná a která jsou k dispozici. V závěru kapitoly je přehledně popsána celá datová struktura.

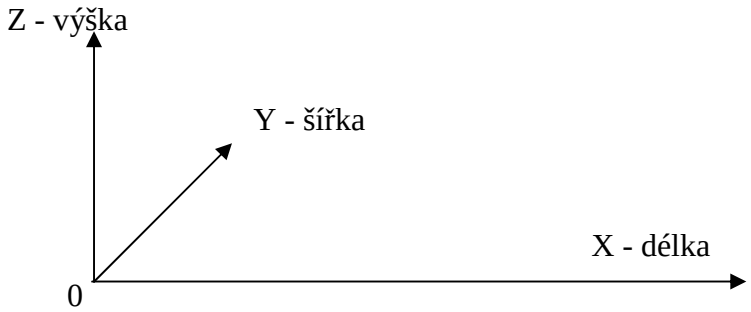
3.1 Definice typu úlohy

Před definicí verbálního modelu řešené úlohy si shrňme její základní typové zařazení:

- obecně tří-dimenzionální BP,
- konvexní a heterogenní objekty i kontejnery tvaru kvádru,
- off-line přístup,
- optimalizační problém (podle definice v kapitole 1.1.2).

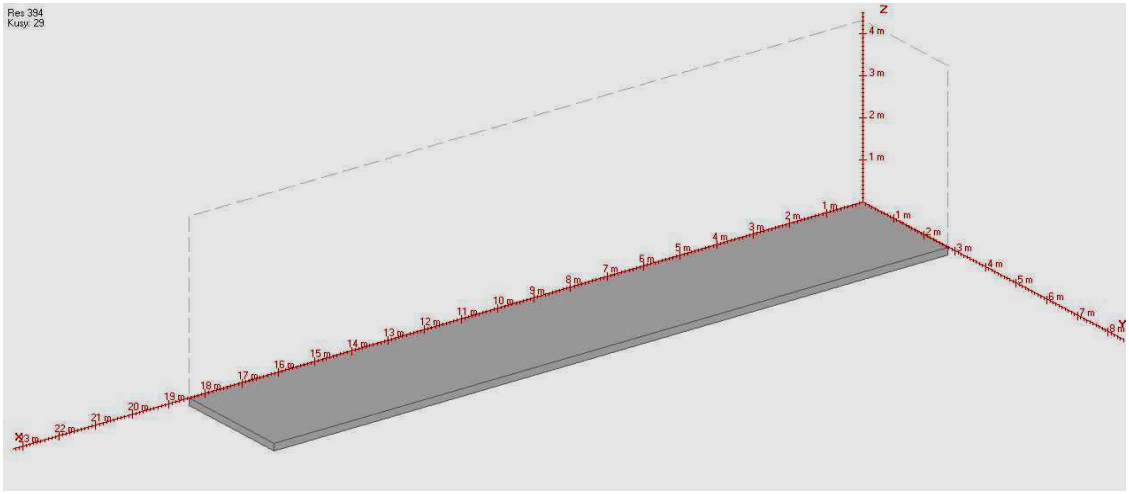
3.1.1 Dimense úlohy

V návaznosti na dělení podle počtu dimensí bude v práci a datovém modelu uvažována praktická prostorová úloha, tedy úloha 3D BP. Zde je pro model velmi důležité umístění středu osových souřadnic v předním levém rohu kontejneru. X-ová souřadnice představuje délku kontejneru, Y-ová šířku a Z-ová výšku.

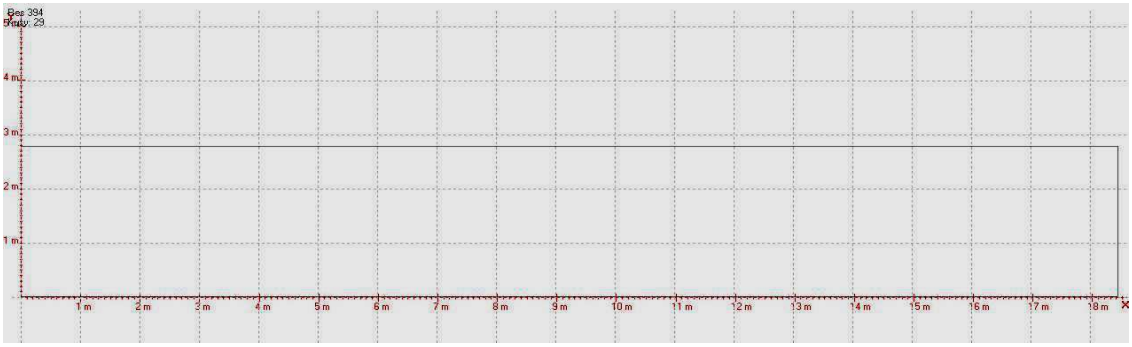


Obrázek 3.1 Schéma umístění souřadnic

Pro jasnější představu uvádím příklad, jak bude v prostoru zaznamenán „kontejner“, zde na obrázcích Obrázek 3.2 a Obrázek 3.3 konkrétně plošinový železniční vůz řady Res:



Obrázek 3.2 Plošinový vůz Res v programu JetLoad (3D)



Obrázek 3.3 Plošinový vůz Res v programu JetLoad (půdorys)

V některých výjimečných situacích však bude z hlediska výpočetní složitosti vhodné omezit se v algoritmech na dvourozměrnou úlohu či dokonce jednorozměrnou úlohu. Dvourozměrný model vznikne redukcí o výšku, tedy Z-ovou souřadnici. Dvourozměrnou úlohu lze tedy použít v případě nakládání objektů, které se nemohou klopit a současně na ně nesmí být stohovány další objekty, a to při splnění podmínky, že se vejdou do kontejneru na výšku.

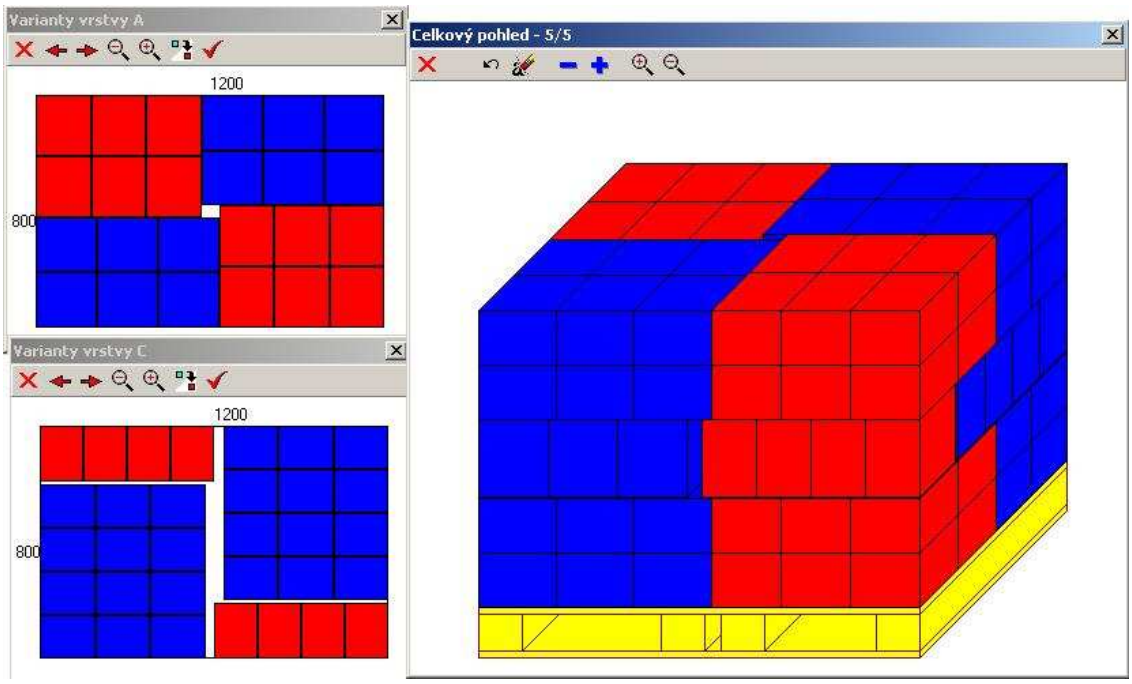
V případě nakládání specielních objektů (například kontejnerů[4] nebo vojenské techniky) na dopravní prostředky lze úlohu zredukovat na větvený přiřazovací problém (vztah n ku 1 = n objektů na jeden dopravní prostředek, kde je počet objektů na jednom dopravním prostředku relativně nízký a jsou sledovány pouze parametry délky, a to jak u nakládaných objektů, tak kontejneru). Tato dílčí úloha bude tedy redukována o dva rozměry, opět o výšku (osa Z) a o šířku (osa Y). Při řešení jednorozměrné úlohy lze použít algoritmus hrubé síly, který pro rozumný počet objektů dosahuje optimálního řešení i v reálném čase.

[4] zde kontejner ve smyslu přepravní jednotky, například ISO 1C

3.1.2 Typy a homogennost objektů a kontejnerů

Z hlediska homogennosti objektů nebude úloha omezena, to znamená, že se bude zabývat nakládáním heterogenních objektů, tedy objektů svými vlastnostmi navzájem různých (především rozměry a hmotností).

Jak bylo řečeno v analýze, homogenní varianta úlohy přináší značné zjednodušení. Lze sestavit rychlý a exaktní algoritmus (v práci pro něj budu používat název „Algoritmus paletizace“), který bude užitečný i při dílčích řešeních obecně heterogenních úloh. To v případě, že část množiny nakládaných objektů je homogenních. Při práci s algoritmy jsem experimentálně jsem zjistil a ověřil, že počet stejných objektů musí být roven nebo větší 4, aby se vyplatilo použít dílčí řešení pomocí algoritmu paletizace. Podrobnosti a popis algoritmu je uveden v kapitole 4.4.1.



Obrázek 3.4 Nakládání homogenních objektů (Pallload)

Co se týče typu nakládaných objektů, budou uvažovány pouze objekty i kontejnery kubických konvexních tvarů, a to obecně kvádrů. V praxi se nezřídka nakládají také objekty cylindrických tvarů (například válce, sudy, roury, atd…). V tomto směru dochází tedy k určité abstrakci od řešení praktických problémů.

Podobně jako homogenností nakládaných objektů se můžeme zabývat homogenností kontejnerů, tedy prostorů, do kterých budou objekty nakládány. V modelu úlohy budeme uvažovat kubické heterogenní kontejnery, jejich výška bude vždy omezena, a to i u otevřených typů prostorů (například plošinový železniční vůz).

3.1.3 Off-line řešení

Z hlediska způsobu vstupu objektů do úlohy bude uvažována úloha deterministická neboli off-line Bin Packing. Jedná se o základní předpoklad, neboť analýza a uspořádání nakládaných objektů před vlastní optimalizací je stěžejní částí disertační práce.

Off-line znamená, že počet a vlastnosti všech objektů jsou známy již před výpočtem a po zahájení výpočtu již nelze množinu zpracovávaných objektů měnit. Naopak on-line Bin Packing předpokládá, že objekty do výpočtu vstupují náhodně až během tohoto výpočtu. I když se z podstaty slova „náhodné“ zdá z hlediska vlastního nakládání řešení on-line problému obtížnější, není tomu tak. Řešení off-line totiž musí uvažovat celou množinu nakládaných objektů, zatímco při on-line problému se při postupném vstupu objektů do úlohy dílčí řešení již nepřepočítává.

3.2 Omezující podmínky

Jak ukázala analýza, oblast respektovaných omezujících podmínek může být velmi rozsáhlá. V modelu řešeném v této disertační práci se pokusím maximálně přiblížit praktickému problému.

Omezující podmínky definují přípustné řešení. V praxi ovšem někdy dochází k vědomému porušení některých omezujících podmínek. Pokud připustíme toto porušení omezujících podmínek i v našem modelu, budou tyto omezující podmínky převedeny na dílčí části hodnotové účelové funkce, viz kapitola 3.3.4. Nejprve však představím všechny možné omezující podmínky. Případnou modifikaci na účelovou funkci pouze zmíním a uvedu dané podmínky a možnosti.

Omezující podmínky lze obecně rozdělit podle jejich působnosti a „místa“ platnosti v průběhu optimalizace. Jejich působnost je řízena hierarchicky (zde od nejnižší váhy po nejvyšší účinnost):

- lokální - platí pro jednotlivé objekty:
  - pro nakládaný objekt,
  - pro nakládání objektu v konkrétní úloze nakládání;
- globální – jsou společné všem objektům, tedy celé úloze nakládání.

Složitost respektování jednotlivých omezujících podmínek se liší především podle této působnosti. Obecně lze říci, že jednodušeji se ošetřuje dodržení lokálních omezení, protože se týkají pouze jednoho objektu nezávisle na ostatních objektech. Zpracování globálních podmínek je výpočtově náročnější, protože zahrnuje testování více objektů a jejich vzájemných poloh.

3.2.1 Přehled omezujících podmínek

OP 1. naložené objekty se musí rozměrově vejít do prostoru kontejneru nebo jinak vymezeného prostoru

OP 2. maximální únosnost kontejneru

OP 3. zatížení náprav, pokud je kontejner dopravním prostředkem

OP 4. rozložení hmotnosti na ploše kontejneru

OP 5. poměr zatížení náprav, pokud je kontejner dopravním prostředkem

OP 6. stabilita naložených objektů z hlediska jejich překlopení

OP 7. stabilita objektů z hlediska jejich uložení

OP 8. objekt se může otáčet

OP 9. objekt se může klopit

OP 10. objekt musí ležet na podlaze, tedy objekt se nesmí stohovat

OP 11. objekt musí být uložen sám na šířku kontejneru

OP 12. na objekt se můžou / nesmí stohovat další objekty (nosnost objektu)

OP 13. definice a umístění skupin objektů v určeném pořadí

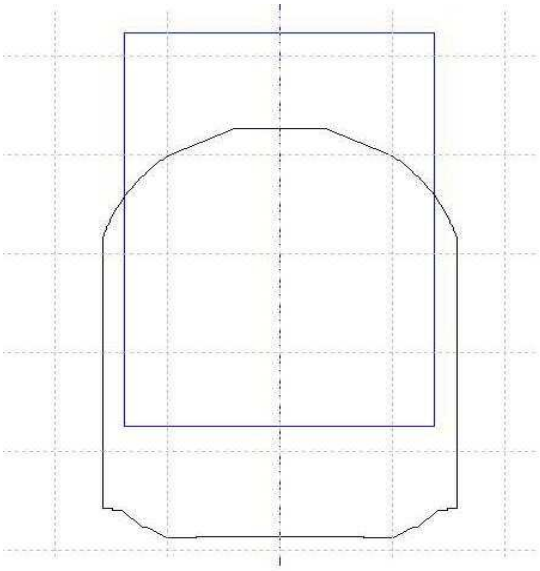
OP 14. umístění objektu z hlediska přepravních podmínek

3.2.2 Rozměrové omezující podmínky (OP 1)

Snad nejdůležitější omezující podmínkou v nakládacích úlohách jsou rozměrové omezující podmínky. Nakládaný objekt nesmí po svém umístění v kontejneru přesáhnout jeho prostorové rozměry. V uzavřeném typu kontejneru je dodržení této omezující podmínky naprosto jasné a v praxi přirozeně vynutitelné. Jinak tomu je u otevřených kontejnerů, například plošinových železničních vozů. V praxi totiž některé objekty mohou přesáhnout rozměry tohoto vozu. V železniční praxi se to nazývá překročení ložné míry (PLM). V modelu lze toto porušení podmínek připustit, v praxi běžně nastává a řeší se. Například v železniční dopravě se v ČR používá pro zjištění prostorové průchodnosti zásilky informační systém Mimoza (viz kapitola 1.8.5).

V navrhovaném modelu lze porušení rozměrových omezujících podmínek připustit. Tento stav však bude penalizován v účelové funkci.

Zde stojí za povšimnutí, že rozměrové omezující podmínky jsou pro člověka poměrně snadno vizuálně ověřitelné. Pro výpočetní techniku se však jedná o složitější problém, zvláště při ověřování prostorové průchodnosti ložných měr.



Obrázek 3.5 Nárys vozu Res a profilu UIC (JetLoad)

Matematický vzorec popisující tuto podmínku uvedu pouze pro jeden rozměr X. Pro rozměry Y a Z by byl analogický.

$$\max_{i=1}^{n}(px_i + pl_i) \leq L \quad ; \qquad px_i \geq 0 \qquad 3.1$$

n	počet objektů
px	umístění objektu v prostor kontejneru vzhledem k ose X
pl	délka objektu (závisí na jeho poloze – otočení / klopení)
L	délka kontejneru

3.2.3 Hmotnostní omezující podmínky

Základní omezující podmínka dodržení únosnosti kontejneru (OP 2) je zřejmá z následujícího vztahu, který udává, že součet hmotností všech naložených objektů nesmí přesáhnout nosnost kontejneru.

$$\sum_{i=1}^{n} b_i \cdot m_i \leq N \qquad 3.2$$

n	počet objektů
b	binární proměnná, která značí, že je objekt naložen v kontejneru
m	hmotnost objektu
N	nosnost kontejneru

Součástí skupiny hmotnostních omezujících podmínek jsou další praktická omezení. V případě, kdy je kontejnerem dopravní prostředek s nápravami (nákladní prostor nákladního auta, železniční vůz), je velmi důležité sledovat maximální zatížení náprav (OP 3). Pro respektování této omezující podmínky je třeba znát umístění

těžiště nakládaného objektu. V tomto modelu proto předpokládám jeho zadání. Pokud není těžiště zadáno, umístí se automaticky do geometrického středu objektu.

Další důležitou podmínkou je pravidelné rozložení hmotnosti obecně v různých osách kontejneru (OP 4). Tedy

- podélné rozložení hmotnosti - ve směru osy X,
- příčné rozložení hmotnosti – ve směru osy Y,
- výškové rozložení hmotnosti, což ovlivňuje stabilitu zásilky.

V modelu budou samozřejmě sledovány všechny tyto osy, tedy (ne)překročení určitého poměru rozložení vůči středovému vyvážení. Z hlediska výškové stability bude tento střed posunut na „podlahu“ kontejneru, tedy hodnota na ose Z = 0.

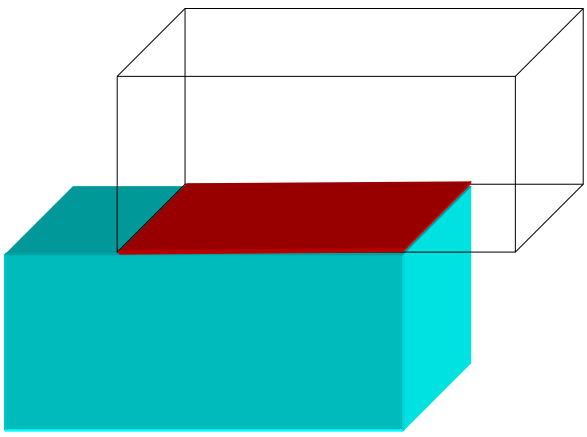
Podobně lze také sledovat poměr zatížení náprav (OP 5), opět pouze v případě, kde je kontejnerem dopravní prostředek s nápravami.

Hmotnostní omezující podmínky bude možno při řešení úlohy porušit, ovšem opět bude toto penalizováno v účelové funkci.

3.2.4 Stabilita objektů

Výšková stabilita (OP 6) souvisí s hmotnostními omezujícími podmínkami. Cílem dodržení této omezující podmínky je minimalizovat riziko nežádoucího překlopení objektu vlivem působení dynamických sil při přepravě. To záleží především na výšce polohy těžiště nakládaného objektu.

Stabilitou objektů z hlediska jejich uložení (OP 7) se rozumí minimalizace rizika sesutí nebo pádu stohovaných objektů. Objekt proto musí spočívat minimálně určeným procentem své plochy na pevné jiném objektu. Toto procento musí být větší než 50%.



Obrázek 3.6 Stabilita stohovaných objektů

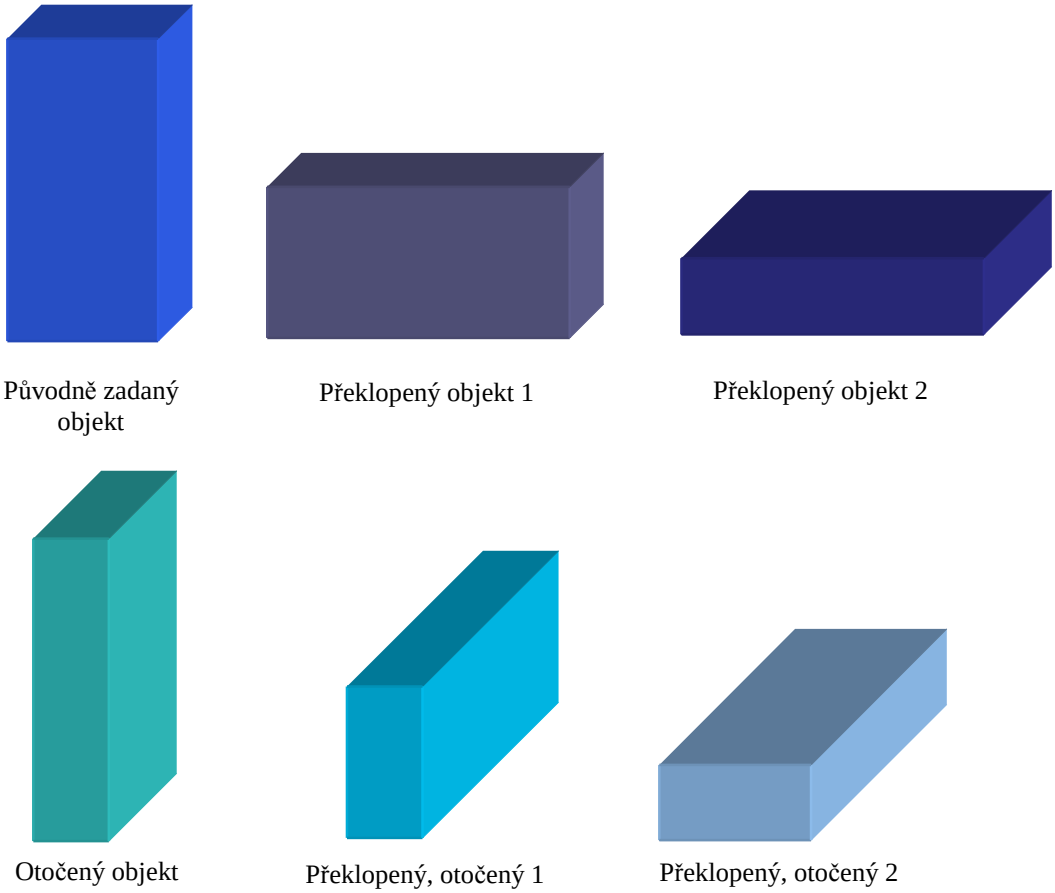
3.2.5 Poloha a umístění objektu

Další častou množinou omezujících podmínek jsou konkrétnější požadavky na umístění objektů.

V první řadě je to možnost manipulace s objekty. Pro každý objekt lze stanovit, kdy je přípustné naložit ho v jiné orientaci, než byl zadán. Manipulace je možná ve dvou rovinách:

- otáčení kolem vertikální osy (OP 8),
- překlápění kolem horizontální osy (OP 9).

Z toho vyplývá celkem 6 možností[5] základních poloh objektu, jak je zřejmé z obrázku Obrázek 3.7. Pro každý objekt by bylo možné umožnit nastavit přípustnost každé z jeho 6 poloh. Toto zpřesnění není ani tak komplikované pro navrhovaný model, ale je příliš náročné na uživatele, který by musel všechny vlastnosti zdlouhavě nastavovat.



Obrázek 3.7 Možné polohy naložení objektu

Další požadavky na umístění objektů vyplývají především z charakteru nakládaného zboží a z požadavků na fixaci a zabezpečení objektů v kontejneru proti

[5] Skutečný počet poloh je dvojnásobný, tedy 12. Vyplývá například z požadavku orientace naloženého zboží ve směru jízdy dopravního prostředku atp. V modelu se s tímto rozšířením nepočítá.

posunutí nebo překlopení. Fixační prvky jsou nejčastěji na podlaze nebo bokách kontejneru. Pro umístění popruhů či jiných fixačních pomůcek je nutné ponechat nějaký volný prostor. Z toho vyplývá potřeba těchto dvou omezujících podmínek:

- objekt musí být na podlaze (OP 10),
- objekt musí být sám na šířku kontejneru (OP 11).

Při nakládání objektů může dojít k potřebě umístit jeden objekt na druhý. V praxi v tomto modelu se používá pojem stohování. Například ve skladech a dopravních prostředcích musí zboží ležet na podlaze, tedy nesmí být stohováno, nesmí se na něj stohovat další zboží, nesmí se stohovat těžší na lehčí zboží atd. Takže podle charakteru nakládaného objektu lze potom z hlediska stohování rozlišovat několik dalších omezujících podmínek:

- zda může být objekt stohován (OP 10); jedná se o inverzní omezující podmínku k výše uvedené podmínce, že objekt musí být na podlaze,
- stohovací nosnost objektu (OP 11) určuje, jaká hmotnost může být na objektu „položena“. Takže při hodnotě „0“ se nesmí na objekt pokládat žádné další objekty. Pokud není možno stohovací nosnost objektu zadat, v modelu je počítáno s tím, že ji lze získat paušálně výpočtem z vlastní hmotnosti objektu pomocí stohovacího koeficientu.

$$Ns = m \cdot k_s \qquad 3.3$$

Ns	stohovací nosnost objektu
m	vlastní hmotnost objektu
ks	stohovací koeficient (v modelu konstanta pro všechny objekty)

Dalším omezením je, že objekt musí být umístěn v uzavřeném kontejneru (krytém dopravním prostředku, voze). Například z důvodu jeho ochrany před povětrnostními vlivy (OP 14).

3.2.6 Priorita a skupiny objektů

Svůj význam může sehrát i priorita objektů či skupin objektů, nebo-li pořadí, ve kterém budou nakládány či vykládány. Tento požadavek je v praxi uplatnitelný při rozvozových úlohách, kde musí být zaručen volný přístup k postupně vykládaným objektům.

Rozdělení objektů do skupin, které pak mají další vlastnosti může být i z jiných důvodů, například některé druhy nebezpečného zboží nemohou být přepravovány ve vzájemné blízkosti či ve společném prostoru.

Proto v modelu respektuji následující omezující podmínky, které s touto problematikou souvisí. Použití těchto podmínek předpokládá vytvoření skupin objektů, které se z tohoto hlediska chovají stejně. Nastavení omezujících podmínek potom tedy platí vždy pro danou skupinu objektů:

- priorita neboli pořadí nakládání / vykládání skupin,
- kolizní matice skupin, vyjadřující, které skupiny mohou / nesmí být nakládány do společného prostoru,
- přímé přiřazení skupiny ke konkrétnímu typu kontejneru (typům kontejnerů), do kterého má být naložena,
- vyjmenování kontejnerů, do kterých nesmí být skupina objektů naložena,
- možnost dokládat do kontejneru ke skupině objekty z jiných skupin,
- přípustnost rozdělit objekty dané skupiny do více kontejnerů.

3.3 Účelová funkce a kritéria úlohy

Účelová funkce slouží k ohodnocení kvality dosaženého řešení. V nakládacích úlohách si lze představit celou řadu různých kritérií, popisující požadovanou kvalitu, a jim odpovídajících účelových funkcí. Vyjmenujme si ty nejčastější z nich:

- maximalizace využití ložného prostoru kontejneru,
- maximalizace využití nosnosti kontejneru,
- maximalizace počtu naložených objektů.

Z praktických zkušeností, které mám s různými potřebami při optimalizaci nakládání jasně vyplývá, že není možné určit jednoznačně jediné kritérium z výše jmenovaných, které může být obecně použito pro optimalizaci nakládky. Existuje totiž celá řada typů úloh a rozdílných požadavků na výsledek jejich řešení. Je zde sice možnost, že by si uživatel vybral kritérium, které nejvíce vyhovuje řešenému typu úlohy. Například software *packVol* (viz kapitola 1.8.2) umožňuje výběr z prvních třech jmenovaných. Ze zkušenosti však vím, že uživatel se příliš neobtěžuje přemýšlením a výběrem vhodného kritéria.

Z těchto důvodů jsem se rozhodl řešit problém jako multikriteriální a použít hodnotovou účelovou funkci. To znamená, že účelová funkce bude obsahovat sice všechny položky kritérií, ale jednotlivá kritéria budou převedena na složky účelové funkce, navíc s modifikací pomocí váhového koeficientu. Tyto koeficienty navíc popíší důležitost, určitou hierarchickou strukturu kritérií.

Stěžejní částí disertační práce bude právě mechanismus k určení správných hodnot těchto váhových koeficientů. Více bude uvedeno v následujících kapitolách.

3.3.1 Přehled kritérií

Kvalitu řešení lze posuzovat podle nejen výše uvedených kritérií. Pro dosažení obecně platných a prakticky použitelných výsledků definujme množinu kritérií, která budou figurovat v účelové funkci navrhovaného modelu:

Krit 1. minimalizace ceny použitých kontejnerů

Krit 2. minimalizace ceny za přepravu

Krit 3. maximalizace užitku naložených objektů

Krit 4. maximalizace využité prostorové kapacity kontejnerů

Krit 5. maximalizace využití nosnosti kontejnerů

Krit 6. minimalizace penalizačních poplatků za nenaložený objekt

Krit 7. minimalizace penalizačních poplatků za porušení omezujících podmínek (pokud je dovoleno)

3.3.2 Cenová kritéria

První skupinou kritérií jsou tzv. cenová kritéria, která ohodnocují řešení podle reálných finančních hodnot.

Za použití každého kontejneru je stanoven nějaký poplatek (Krit 1). V praxi si pod tím lze představit náklady spojení s objednáním a/nebo náklady na přistavení dopravního prostředku. Na základě tohoto kritéria může algoritmus naložit raději dva levné kontejnery, i když nebude plně využita jejich kapacita, než jeden drahý. Matematicky lze toto kritérium popsat následujícím vztahem:

$$\min \rightarrow \sum_{k=1}^{K} C_k \qquad 3.4$$

K	počet použitých kontejnerů
C	cena za použití kontejneru

Dalším cenovým kritériem je minimalizace ceny za přepravu (Krit 2), pokud se nějaká realizuje. Náklady na přepravu naloženého nákladu vyplývají z použitého druhu dopravy, z ujeté vzdálenosti, a pravidel cenotvorby. Model úlohy počítá s poměrně robustním modulem cenotvorby[6], který slouží k výpočtu přepravních nákladů. Z tohoto důvodu zde neuvádím matematické vyjádření tohoto kritéria.

Dalším, pseudo-cenovým kritériem je maximalizace užitku z naložených objektů (Krit 3). Užitek neboli cena objektu definuje důležitost neboli hodnotu objektu pro nakládku. Na základě tohoto parametru algoritmus naloží přednostně objekty s větší hodnotou, pokud nebude možno naložit všechny objekty. Kritérium bude bráno v potaz pouze pokud mají jednotlivé objekty zadané různé hodnoty užitku. Výchozí hodnota je pro všechny objekty stejná. Toto kritérium odpovídá hlavnímu kritériu v příbuzné úloze batohu (kapitola 1.7.1).

$$\max \rightarrow \sum_{i=1}^{n} b_i \cdot c_i \qquad 3.5$$

[6] Pro účely cenotvorby je možné navrhovaný model a algoritmy nakládky doplnit vhodným trasovacím modulem. Nicméně toto téma se již příliš vzdaluje vytyčenému tématu disertační práce, proto se spokojíme se zadanou přepravní vzdáleností.

n	celkový počet objektů
b	binární proměnná, která značí, že je objekt naložen v kontejneru
c	cena, hodnota užitku objektu

3.3.3 Kvalitativní kritéria

Kvalitativní kritéria oceňují kvalitu získaného řešení. Lze jich navrhnout celou řadu, ale pro účely modelu se spokojíme s jejich základní skupinou.

Nejzřejmější je kritérium sledující maximální využití prostorové (objemové) kapacity kontejnerů (Krit 4). Zde vstupuje v poměr zabraný objem všech v kontejneru naložených objektů ku celkovému objemu kontejneru.

$$\max \rightarrow \frac{\sum_{i=1}^{n} b_i \cdot v_i}{V} \qquad 3.6$$

n	celkový počet objektů
b	binární proměnná, která značí, že je objekt naložen v kontejneru
V	objem kontejneru
v	objem objektu

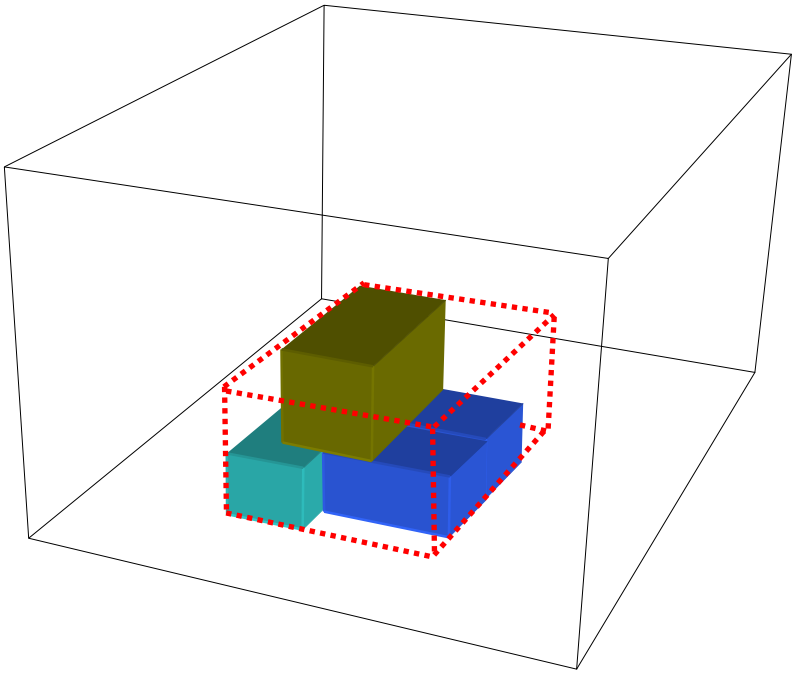
V případě nakládání objektů do „neohraničených“ prostorů (např. sklad) nelze použít nebo je nevhodné použít ukazatel využití objemové kapacity. Pro tyto účely definujme kritérium minimalizace záboru konvexního ohraničujícího prostoru, které vyjadřuje jakousi kompaktnost naloženého celku. Matematické vyjádření tohoto kritéria je již poněkud komplikované:

$$\min \rightarrow \left(\underset{i=1}{\overset{n}{Max}}(px_i + pl_i^x) - \underset{i=1}{\overset{n}{Min}}(px_i) \right) \cdot \left(\underset{i=1}{\overset{n}{Max}}(py_i + pl_i^y) - \underset{i=1}{\overset{n}{Min}}(py_i \right) \cdot \cdot \left(\underset{i=1}{\overset{n}{Max}}(pz_i + pl_i^z) - \underset{i=1}{\overset{n}{Min}}(pz_i) \right) \qquad 3.7$$

$$px_i \geq 0$$

n	počet objektů
px	umístění objektu v prostor kontejneru vzhledem k ose X
py	umístění objektu v prostor kontejneru vzhledem k ose Y
pz	umístění objektu v prostor kontejneru vzhledem k ose Z
plx	délka objektu (závisí na jeho poloze – otočení / klopení)
ply	šířka objektu (závisí na jeho poloze – otočení / klopení)
plz	výška objektu (závisí na jeho poloze – otočení / klopení)

Zřejmé je naopak jeho grafické vyjádření tohoto kritéria, kde červený kvádr představuje minimalizovaný prostor:



Obrázek 3.8 Konvexní ohraničení naložených objektů

Dalším kritériem je maximalizace využití nosnosti kontejnerů. Matematický vztah, popisující toto kritérium, je analogický s kritériem maximalizace využití objemové kapacity:

$$\max \rightarrow \frac{\sum_{i=1}^{n} b_i \cdot m_i}{N} \qquad 3.8$$

n	celkový počet objektů
b	binární proměnná, která značí, že je objekt naložen v kontejneru
N	nosnost kontejneru
m	hmotnost objektu

3.3.4 Penalizační kritéria

Pokud model nepracuje s kritériem maximalizace užitku, je nutné do účelové funkce nějak zapojit kritérium postihující to, zda byly všechny objekty naloženy. Tímto kritériem je minimalizace penalizačních poplatků za nenaložený objekt (Krit 6). Kritérium je také nutné použít ve spojení s kritériem minimalizace následných přepravních nákladů (Krit 2). To z toho důvodu, že minimální náklady na přepravu by byly tehdy, pokud by se žádné zboží nepřepravovalo. Této situaci se musí algoritmus řešení vyvarovat, proto jako protiváha vstupuje do účelové funkce kritérium minimalizace poplatků za nenaložené objekty.

V kapitole 3.2 o omezujících podmínkách bylo řečeno, že v praxi je v některých případech dovoleno omezující podmínky překročit. Tyto situace skutečně nastávají. V modelu to však musí být postihnuto v účelové funkci, a to pomocí sady penalizačních kritérií – poplatků za porušení omezujících podmínek (Krit 7).

Povolené porušení omezujících podmínek nesmí mít negativní vliv na bezpečnost uložení zboží. Většinou pouze omezuje dopravní podmínky. Jedná se tedy především o:

- Rozměrové omezující podmínky, kdy část naloženého objektu přesahuje definovaný prostor kontejneru. Tímto kontejnerem tedy může být pouze otevřený dopravní nebo přepravní prostředek, například plošinový vůz, paleta, atd.
- Hmotnostní omezující podmínky – překročení nosnosti dopravního nebo přepravního prostředku, větší než povolený rozdíl zatížení náprav, atd.

Překročení omezujících podmínek není neomezené. Pokud je vůbec povoleno, může se pohybovat do stanovené hranice dané buď absolutní číselnou hodnotou nebo procentuelně. Například nosnost nějakého dopravního prostředku může být překročena maximálně o 10%. Z toho také bude vyplývat maximální možná hodnota penalizace.

Výpočet penalizačních poplatků se může řídit několika způsoby závislými na průběhu oceňovací funkce:

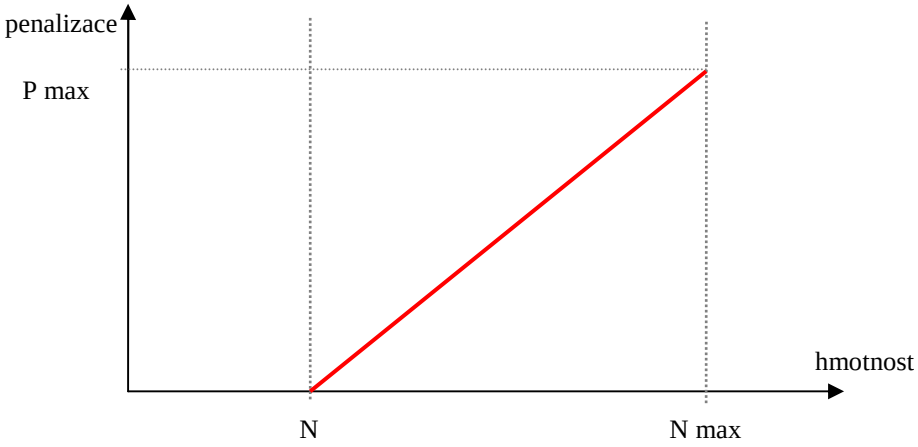
- lineární,
- exponenciální,
- sigmoidní.

Při nejjednodušším lineárním penalizačním postihu se penále vypočítá podle lineární funkce. Následující příklad uvádí výpočet penalizace za překročení omezující hmotnosti[7]. Ostatní výpočty penalizací budou analogické.

$$\min \rightarrow P_{ph} = \left| \sum_{i=1}^{n} b_i \cdot m_i - N \right| \cdot p_h \qquad \sum_{i=1}^{n} m_i \leq N \cdot \left(1 + \frac{k_{ph}}{100} \right) \qquad 3.9$$

Pph	penalizace za překročení nosnosti kontejneru
n	celkový počet objektů
b	binární proměnná, která značí, že je objekt naložen v kontejneru
m	hmotnost objektu
N	nosnost kontejneru
ph	penalizace za překročení nosnosti o jednotku
kph	procentuelní koeficient maximálního překročení nosnosti

[7] Všechny uvedené vzorce předpokládají, že k překročení nosnosti kontejneru skutečně došlo.



Graf 3.1 Penalizace – lineární průběh

Příklad:

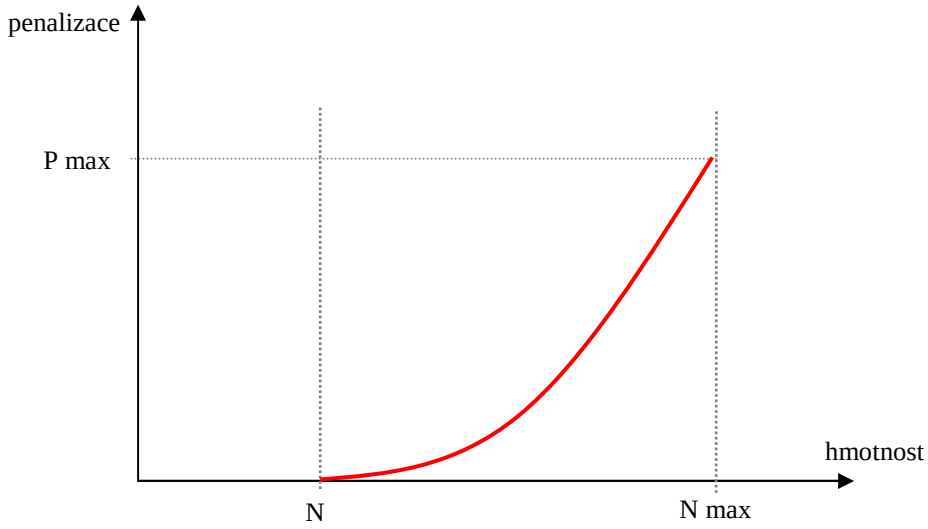
Nosnost kontejneru N je 1000 kg. Koeficient maximálního překročení nosnosti k_{ph} je 10%. Penalizace p_h za překročení nosnosti o jeden kg je 10 cenových jednotek (cj). Pokud bude hmotnost naloženého zboží rovna 1200 kg, je toto naložení nepřípustné, protože překračuje nosnost kontejneru o 20%. Pokud bude hmotnost naloženého zboží rovna 1050 kg, bude penalizováno překročení nosnosti o 50 kg. Podle lineárního vztahu vyjde penalizace P_{ph} na 500 cj.

Exponenciální přístup k penalizaci zohledňuje větší náročnost vypořádání se s důsledky překročení omezující podmínky s rostoucí mírou tohoto překročení.

Opět uvádím vztah pouze pro penalizaci překroční nosnosti kontejneru:

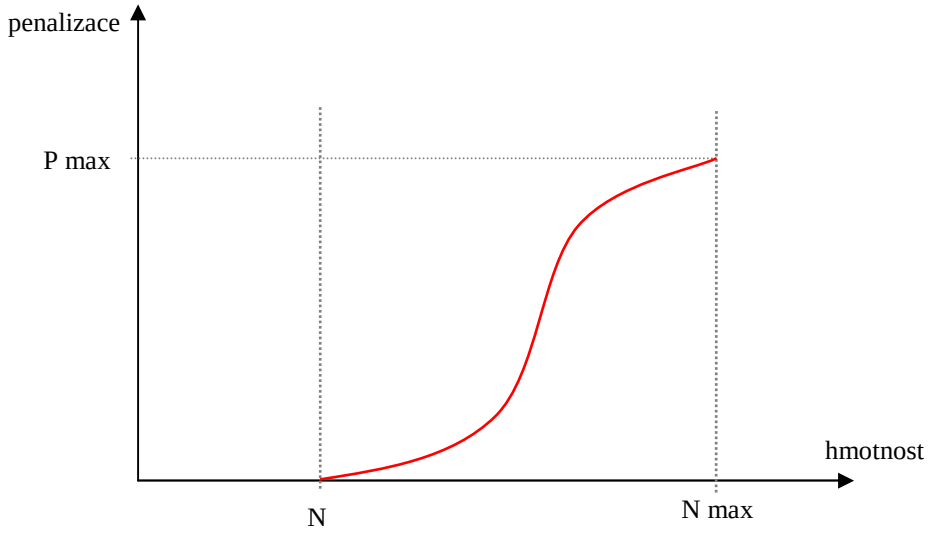
$$\min \rightarrow P_{ph} = p_h^{\left(k_x \cdot \left| \sum_{i=1}^{n} b_i \cdot m_i - N \right|\right)} \qquad \sum_{i=1}^{n} m_i \leq N \cdot \left(1 + \frac{k_{ph}}{100}\right) \qquad 3.10$$

Pph	penalizace za překročení nosnosti kontejneru
n	celkový počet objektů
b	binární proměnná, která značí, že je objekt naložen v kontejneru
m	hmotnost objektu
N	nosnost kontejneru
ph	penalizace za překročení nosnosti o jednotku
kph	procentuelní koeficient maximálního překročení nosnosti
kx	koeficient určující míru působení exponenciální funkce



Graf 3.2 Penalizace – exponenciální průběh

Vyčíslení penalizačních poplatků, které nejvíce odpovídá reálným nákladům na vypořádání se s porušením omezujících podmínek je pomocí sigmoidní funkce. Sigmoida (říká se jí též logistická funkce nebo logistická křivka) je funkce, modelující růst nějaké veličiny. V počáteční fázi je růst přibližně exponenciální, později s rostoucím nasycením se zpomaluje, až se nakonec asymptoticky zastaví.



Graf 3.3 Penalizace – sigmoidní průběh

Penalizaci za překroční nosnosti lze pomocí sigmoidní funkce vypočítat pomocí vztahu:

$$\min \rightarrow P_{ph} = \frac{1}{1 + e^{-\left|\sum_{i=1}^{n} b_i \cdot m_i - N\right|}} \qquad 3.11$$

3.3.5 Účelová funkce

Vzhledem k výše uvedeným předpokladům bude mít účelová funkce v definovaném modelu strukturu složenou z jednotlivých kritérií. Dejme tomu, že extrém hodnotové účelové funkce bude hledán v minimu. Proto bude třeba kritéria, která jsou ze své podstaty maximalizační převézt na kritéria minimalizační.

Obecně bude mít účelová funkce tento tvar:

$$U = \sum_{k=1}^{pk} x_k \cdot \mu_k \cdot C_k \cdot w_k \qquad 3.12$$

U	hodnota celkové účelové funkce
pk	počet kritérií
x	binární proměnná značící, zda bude kritérium zahrnuto do účelové funkce
μ	znaménková funkce - příznak převodu minimalizačního kritéria na maximalizační (1 nebo -1)
C	cenová hodnota kritéria
w	koeficient, který upravuje váhu (důležitost) kritéria v konkrétním výpočtu

Sice by se dalo dosáhnout zjednodušením obecného vzorce sloučením binární proměnné *x* s váhovým koeficientem *w*:

$w = 0$ pro $x = 0$

$w > 0$ pro $x = 1$

Ale dále v řešení bude důležité uchovávat váhy jednotlivých kritérií a odděleně přitom řídit přítomnost kritéria v konkrétním příkladě.

3.4 Požadované výstupy

Pro sestavení modelu je potřeba znát také požadované výstupy řešení. Výstupy řešení mohou nabývat různých podob v závislosti na tom, jak bude s výsledky dále nakládáno.

V praktických nakládacích úlohách je očekáván výsledek v podobě tzv. ložného plánu. Ložný plán představuje popis umístění objektů v konkrétních kontejnerech. Naopak pro akademické účely, například pro porovnání kvality řešení dosažených různými výpočetními algoritmy postačí pro základní orientaci některé popisné charakteristiky, například procentuelní využití ložného prostoru nebo hodnota účelové funkce.

Rozborem možných výstupů a prezentací řešení se podrobně zabývám níže v kapitole 4.6.

3.5 Datová struktura

Popis datových struktur se nebude vzhledem k jejich značnému rozsahu týkat všech datových objektů a jejich vlastností, které se při řešení úlohy používají. Pro lepší přehled rozdělím celkový datový model do několika oblastí. Pro každou oblast uvedu:

- stručný popis,
- lineární model základních datových struktur, včetně popisu datových typů,
- relační model, který popisuje vazby mezi datovými objekty (DB tabulkami).

Relační databázový model sdružuje data do tzv. relací (tabulek), které tvoří základ relační databáze. Tabulka je struktura záznamů s pevně stanovenými položkami (sloupci - atributy). Každý sloupec má definován jednoznačný název, typ a rozsah. Pokud jsou v různých tabulkách sloupce stejného typu, pak tyto sloupce mohou vytvářet vazby mezi jednotlivými tabulkami. Tabulky se poté naplňují vlastním obsahem - konkrétními daty. Relační model je vždy patřičně okomentován, aby bylo zřejmé, jaké datové údaje spolu přímo i nepřímo souvisejí.

3.5.1 Typy atributů

Na tomto místě musím ještě upozornit na jisté obecné vlastnosti a rozdělení atributů. Druh atributu bude důležitý při vlastním řešení při analýze vstupujících dat, kdy se každý typ zpracovává jiným způsobem.

Každý datový typ je tedy definován množinou svých atributů. Ty mohou být v zásadě dvou typů:

- kategoriální (diskrétní),
- spojité (numerické).

Kategoriální typy lze dále členit na:

- binární nabývající pouze hodnot ano/ne,
- nominální nabývající jedné z konečného počtu hodnot, které nejsou vzájemně uspořádány,
- ordinální nabývající jedné z konečného počtu navzájem uspořádaných hodnot.

Příklad:
Hmotnost nebo rozměry jsou atributy spojitého typu. Kategoriální binární atribut je například vlastnost objektu, zda ho lze otáčet nebo stohovat. Určení konkrétní polohy, ve které může být objekt naložen je kategoriální nominální atribut.

3.5.2 Obchodní případ a varianty nakládání

Obchodní případ je jedna konkrétní úloha k nakládání. Svůj název dostal z předpokladu dalšího budoucího rozšiřování modelu, takže obsahuje nejen údaje potřebné pro nakládání, ale i údaje obchodního charakteru, informace o případné

dopravě či přepravě a další popisné informace. Nejdůležitější však je, že obsahuje seznam objektů k naložení a seznam nakládacích variant.

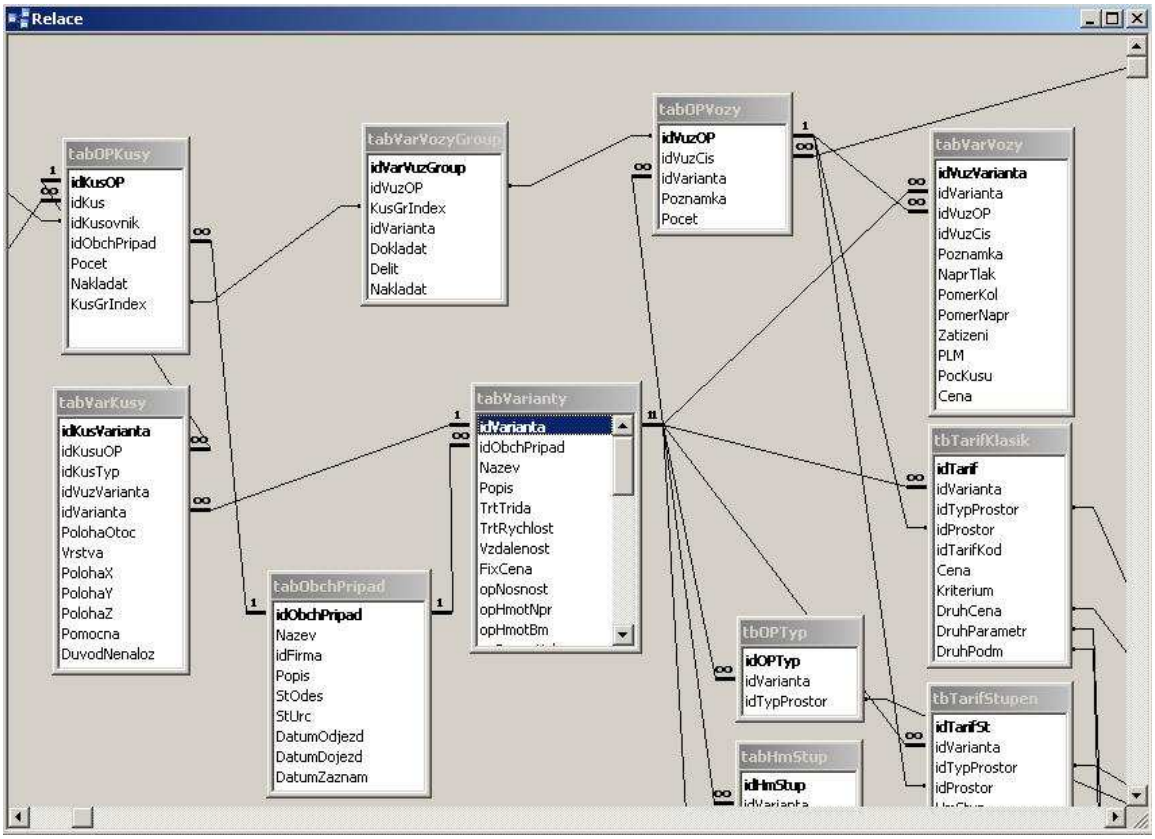
Obchodní případ

- Identifikační číslo
- Název
- Popis
- Odkaz na spolupracující subjekty (zákazník, dopravce, atd)
- Odesílací dopravní bod
- Dopravní bod určení
- Datum realizace
- \+ Seznam nakládaného zboží[8]
- \+ Seznam nakládacích variant

Varianta nakládání

- Identifikační číslo
- Název
- Popis
- \+ Specifikace případné přepravy
    - Druh dopravy
    - Vzdálenost
    - Rychlost
    - Traťová třída
- \+ Nastavení globálních omezujících podmínek
- \+ Seznam kontejnerů, na které lze nakládat
- \+ Nastavení způsobu výpočtu a algoritmu
- \+ Nastavení parametrů analýzy, výpočtů, atd.
- \+ Výsledky nakládání

[8] odrážka „+“ značí netriviální datový typ



Obrázek 3.9 Relační model – Obchodní případ a varianty

Z relačního modelu vyplývá propojení jednotlivých tabulek. Nejdůležitější vztahy tabulky *tabObchPripad* (obchodní případ) jsou:

- vztah k nakládacím variantám (*tabVarianty*) 1:N,
- vztah k objektům vstupujících do úlohy nakládání (*tabOPKusy*) 1:N.

Tabulka *tabVarianty* (nakládací varianty) je svázaná především s tabulkami popisujícími výsledek nakládání:

- tabulka *tabVarKusy* popisující umístění objektů v kontejnerech,
- tabulka *tabVarVozy* popisující seznam použitých kontejnerů.

3.5.3 Nakládané objekty

Datové struktury popisující typy a vlastnosti jednotlivých objektů jsou v tabulkách *tabCisKusy*, *tabKusyTyp* a *tabKusyPodTyp*. Ostatní tabulky jsou pomocné, pro snadné zadávání objektů do úlohy nakládání.

Nejdůležitější je základní popis objektu v číselníku - tabulce *tabCisKusy*, jejíž lineární model vypadá takto:

- Identifikační číslo
- Typová identifikace

- Název
- Popis
- Cena (užitek)
- Barva (pro grafické vyjádření)
- Rozměry
    - Délka (X)
    - Šířka (Y)
    - Výška (Z)
- Hmotnost
- Poloha těžiště (Tx, Ty, Tz)

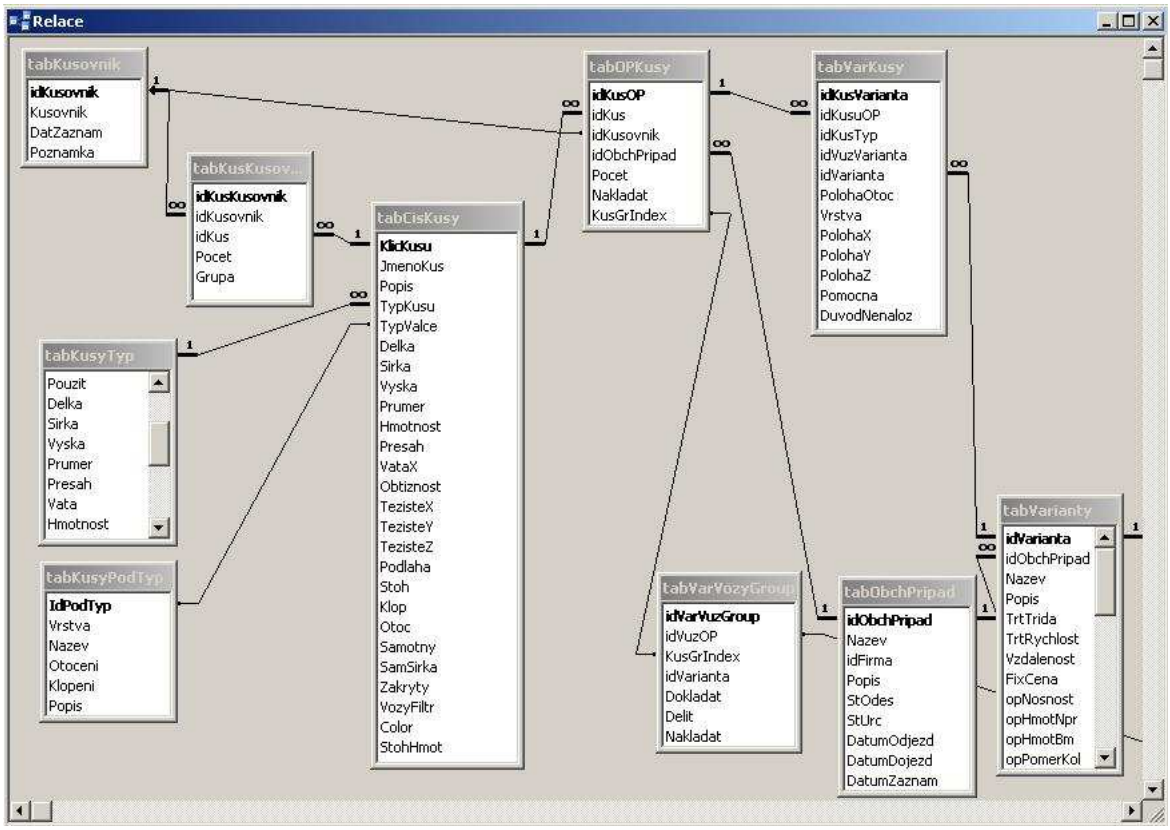
+ Specifikace lokálních omezujících podmínek
    - Lze otáčet
    - Lze klopit
    - Lze stohovat / musí být na podlaze
    - Stohovací hmotnost
    - Maximální přesah (%)
    - Sám na šířku kontejneru
    - V zakrytém kontejneru

Popis výsledku z hlediska naloženého objektu je v další důležité tabulce *tabVarKusy*:

- Identifikační číslo objektu z tabulky *tabCisKusy*
- Odkaz na kontejner, ve kterém je objekt naložen
- Souřadnice umístění objektu v kontejneru (Px, Py, Pz)
- Konkrétní poloha objektu vyplývající z možnosti objekt otáčet nebo klopit
- Specifikace vrstvy, ve které objekt leží, pokud je stohován

+ Případná specifikace důvodu nenaložení objektu do žádného kontejneru

Relační model přehledně popisuje propojení tabulek v oblasti nakládaných objektů.



Obrázek 3.10 Relační model – nakládané objekty

Propojení na pomocné tabulky *tabKusovnik* a *tabKusKusovnik* usnadňuje uživateli výběr a editaci objektů v obchodním případu. Tabulky *tabKusyTyp* a *tabKusyPodTyp* upřesňují typové zařazení objektů, nastavení jejich výchozích parametrů a přípustnost některých lokálních omezujících podmínek. Stěžejní je propojení tabulky *tabCisKusy* na tabulku *tabOPKusy*, které popisuje zařazení objektů do nakládací úlohy (obchodního případu). V tabulce *tabOPKusy* lze nastavit:

- počet jednotlivých objektů z číselníku *tabCisKusy*,
- dočasné vynechání objektu z optimalizace,
- zařazení objektu do skupin aj.

Skupiny objektů jsou popsány v tabulce *tabVarVozyGroup* tak, jak bylo uvedeno v kapitole popisující omezující podmínky spojené s prioritami skupinami objektů. Nejdůležitější parametry jsou:

- propojení do tabulky *tabOPVuz* na konkrétní kontejner,
- nastavení možnosti „dělit skupinu“,
- nastavení možnosti „dokládat skupinu“,
- dočasné vynechání celé skupiny z optimalizace.

3.5.4 Kontejnery

Typy kontejnerů jsou popsány v tabulce *tbTypProstor*:

- Identifikační číslo
- Název typu
+ Specifikace vlastností typu
    - Vlastní hmotnost
    - Počet náprav (pokud je kontejner dopravním prostředkem)
    - Orientace (záleží na tom, co je přední a zadní část kontejneru)
    - Krytý prostor
+ Specifikace použití omezujících podmínek
    - Kontrolovat rozložení hmotnosti
    - Kontrolovat podélné zatížení náprav
    - Kontrolovat příčné zatížení náprav
+ Specifikace tarifních podmínek
    - Tarif pro hmotnostní stupně
    - Tarif pro objemové stupně

Pro příklad uvedu seznam typů, které jsou přednastaveny:

- otevřený prostor,
- obecný omezený prostor,
- sklad,
- kontejner ve smyslu přepravní jednotky,
- paleta a podobné přepravní jednotky,
- loď, lodní prostor,
- silniční vozidlo dvounápravové,
- silniční vozidlo třínápravové,
- návěs silničního vozidla,
- přívěs silničního vozidla,
- železniční vůz dvounápravový,
- železniční vůz čtyřnápravový,
- atd.

Konkrétní položky číselníku „kontejnerů“ jsou v tabulce *tbProstory*. Uváděné vlastnosti jsou tyto:

- Identifikační číslo kontejneru
- Název
- Typ kontejneru (viz *tbTypProstor*)
- Režim (jen železniční vozy)
- Cena, ohodnocení dostupnosti
- Délka (X)
- Šířka (Y)
- Výška (Z)
- Otevřený prostor (lze překročit rozměry)
- Odstranitelnost bočnic
- Odstranitelnost čela
- Odstranitelnost stropu
- Nosnost
- Počet náprav
- Rozvor náprav (prostřední část)
- Rozchod

+ Specifikace omezujících podmínek
    - Maximální poměr podélné rozložení hmotností
    - Maximální poměr příčné rozložení hmotností
    - Maximální výška těžiště

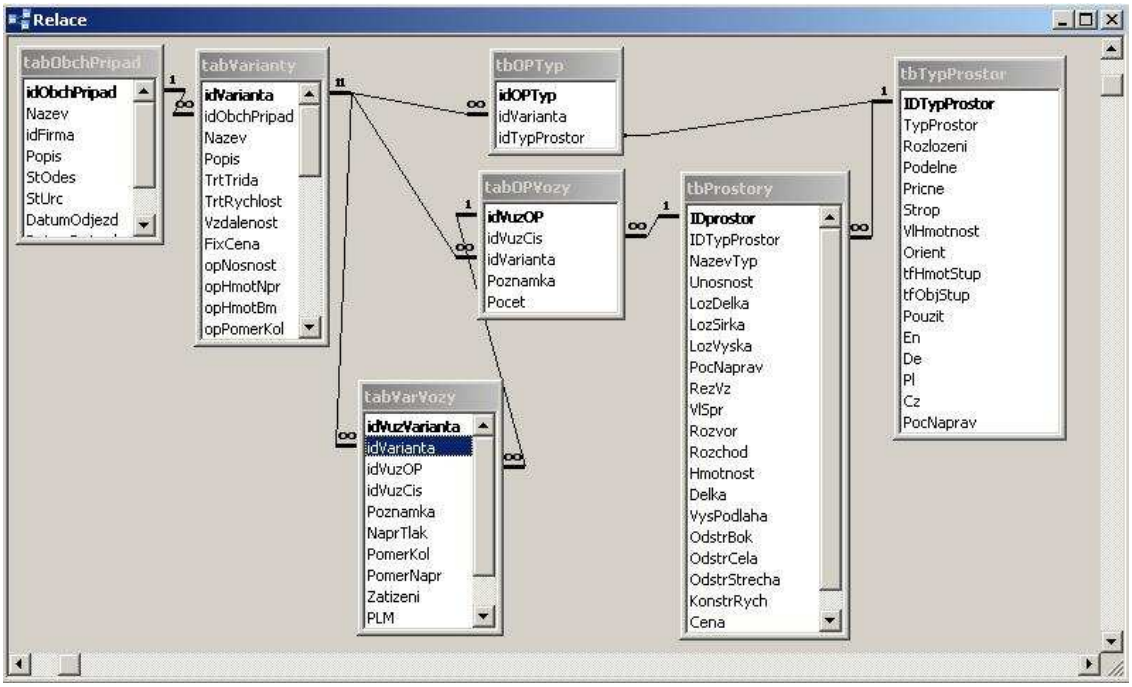
Do nakládací úlohy lze přiřadit buď celou skupinu kontejnerů specifikovanou pouze jejich typem (viz tabulka *tbOPTyp*) nebo lze přiřadit detailně jednotlivé kontejnery. To je popsáno v tabulce *tabOPVozy*:

- Odkaz do číselníku kontejnerů
- Poznámka
- Disponibilní počet kontejnerů
- Cena za použití kontejneru (přistavení dopravního prostředku)

Výsledky nakládání a informace o použitých kontejner jsou v tabulce *tabVarVozy*:

- Odkaz do číselníku kontejnerů

- Celková hmotnost naložených objektů
- Počet naložených objektů
- Cena podle účelové funkce
- Nápravový tlak (pokud jsou nápravy)
- Poměr zatížení kol (příčné rozložení hmotnosti)
- Poměr zatížení náprav (podélné rozložení hmotnosti)
- Je překročená ložná míra

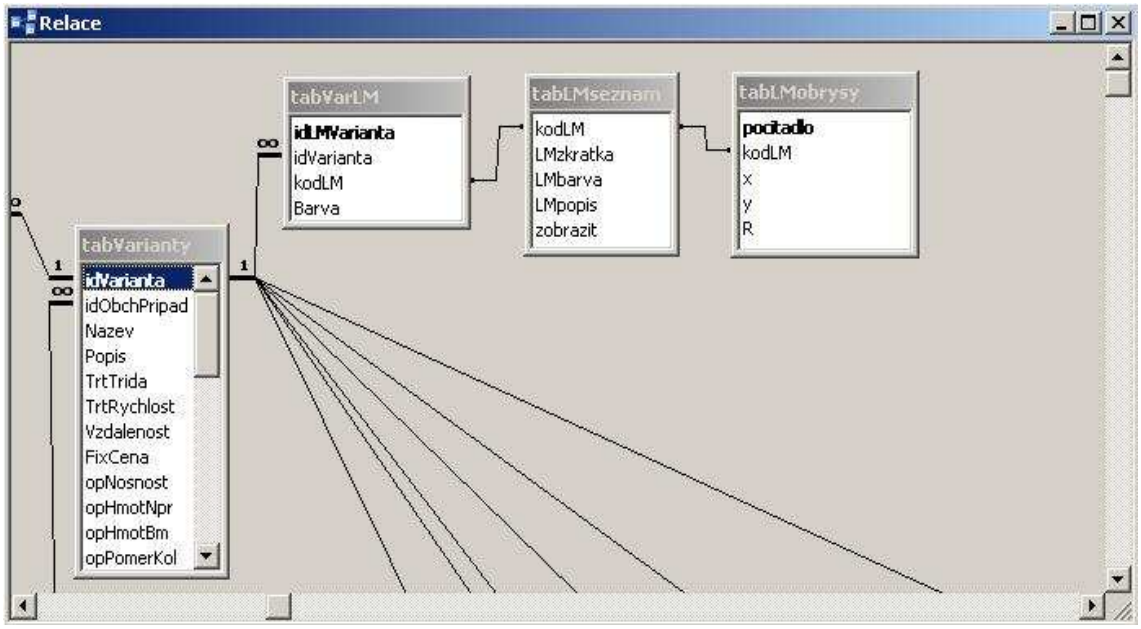


Obrázek 3.11 Relační model – prostory (dopravní a přepravní prostředky)

Relační vztahy mezi popisovanými tabulkami jsou opět zobrazeny a jsou zřejmé z relačního modelu na obrázku Obrázek 3.11 Relační model – prostory (dopravní a přepravní prostředky).

3.5.5 Ložné míry

Způsob záznamu ložných měr v databázi je dostatečně výstižný pouze z relačního modelu.



Obrázek 3.12 Relační model – ložné míry

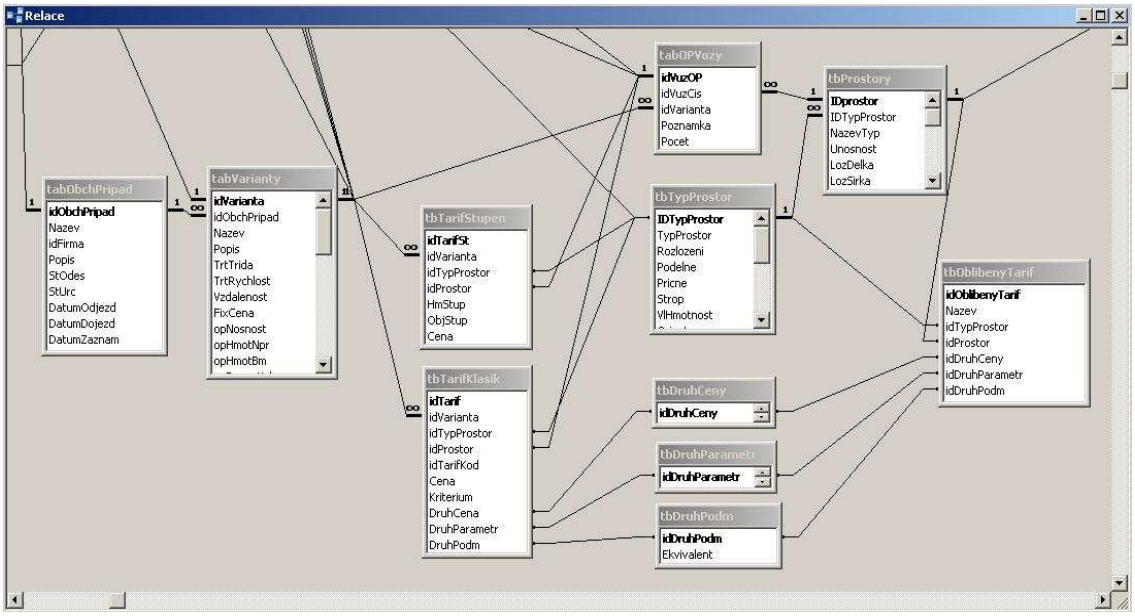
3.5.6 Cenotvorba a tarify

Pro výpočet ceny přepravních nákladů obsahuje model vlastní modul nákladového kalkulátoru, který tuto problematiku řeší. Modul je poměrně složitý, a protože jeho vnitřní mechanismy přímo nesouvisí s tématem této disertační práce, popíši ho pouze stručně.

Nákladový kalkulátor pracuje kromě hmotnosti naložených objektů v kontejneru s:

- přepravní vzdáleností,
- tarifními hodnotami,
- hmotnostními stupni,
- objemovými stupni,
- systémem slev,
- atd.

O složitosti nákladového kalkulátoru jistě přesvědčí snímek Obrázek 3.13 jeho relačního modelu.



Obrázek 3.13 Relační model – tarify a cenotvorba

4 Vlastní řešení

Ze závěrů analýzy současného vědeckého poznání a z popisu navrhovaného modelu úlohy vyplynuly základní požadavky na řešení úlohy. Řešení by mělo být:

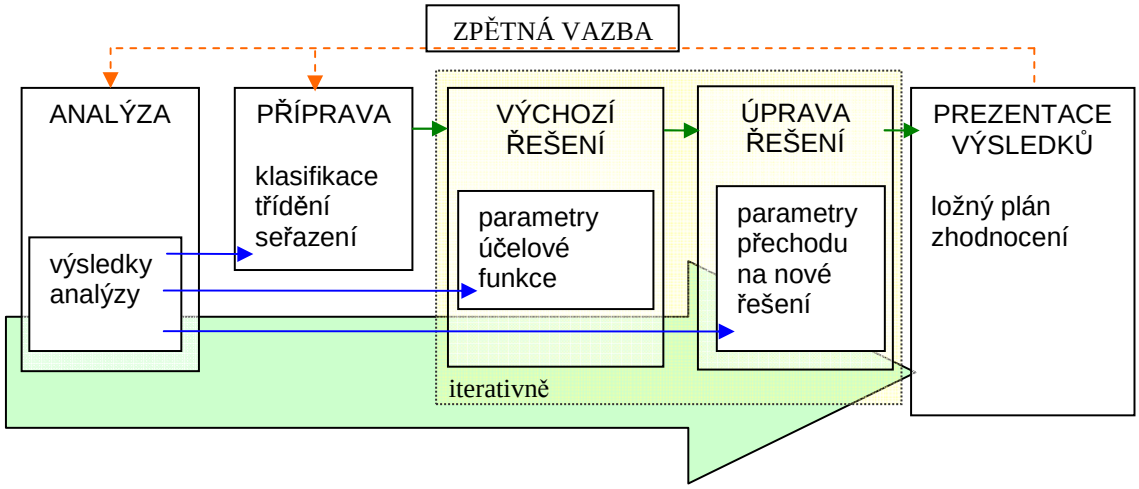
- obecné,
- rychlé,
- automatizované,
- pro uživatele jednoduché a přehledné.

Cílem je připravit vhodný postup řešení, sestavit algoritmy a metody pro řešení problematiky úloh Bin Packingu, které by splňovaly stanovené požadavky. Algoritmy a metody budou testovány na softwarové aplikaci, která je součástí disertační práce. Popis této aplikace a rozbor některých příkladů optimalizace bude uveden v kapitole 6.3.1.

4.1 Přístup k řešení

Splnění stanovených požadavků vyžaduje zcela nový, rozšířený přístup k řešení. Tento přístup lze popsat následujícími fázemi obecného postupu:

1. analýza vstupních dat:
   1.1. analýza nakládaných objektů,
   1.2. analýza disponibilních kontejnerů,
   1.3. analýza omezujících podmínek,
   1.4. určení typu úlohy, určení optimalizačních parametrů,
2. na základě předchozí analýzy příprava vstupních dat:
   2.1. třídění nakládaných objektů,
   2.2. seřazení nakládaných objektů,
   2.3. seřazení kontejnerů,
   2.4. sestava vhodné účelové funkce,
   2.5. příprava mechanismu tvorby variant,
3. postupná tvorba variant řešení, zapamatování si slibných řešení:
   3.1. výpočet výchozího řešení,
   3.2. úprava výchozího řešení,
4. prezentace výsledků nejlepšího řešení,
5. možnost zpětnovazebního mechanismu pro učení algoritmů analýzy a přípravy dat.



Obrázek 4.1 Vliv vstupní analýzy na parametry výpočtu

4.1.1 Data mining

Jistou inspiraci pro výše uvedený postup řešení mi poskytly principy dobývání znalostí z databází a Data Miningu (1). Data Mining je analytická metodologie pro netriviální získávání implicitních, skrytých, dříve neznámých a potenciálně užitečných informací ze strukturovaných dat. Zpočátku se jednalo například jen o využívání regresní analýzy s automatickým výběrem proměnných či prvních rozhodovacích stromů. Další rozvoj statistických metod, databázových aplikací a umělé inteligence spolu s rychlým růstem rychlosti a paměti počítačů jsou novou výzvou pro systematické a plnohodnotné využití data miningové metodologie. Ovšem na rozdíl od „prostého“ využití statistických metod se v procesu dobývání znalostí klade důraz i na přípravu dat pro analýzu a na interpretaci výsledných znalostí.

Data Mining se ve stále větší míře aplikuje nejen ve vědě a akademické sféře, ale dochází k jeho širokému použití v komerční praxi. Časté aplikace jsou především v oblastech marketingu, finančnictví, maloobchodního prodeje, telekomunikací, internetového prodeje, atd.

Protože Data Mining zahrnuje velkou šíři metod a způsobů práce, je obtížné podat jednoznačný návod k jeho postupům. Společnou podstatou všech metodologií je posloupnost několika kroků, jejichž použití může být i iterativní:

1. formulace úlohy a porozumění problému,
2. porozumění datům – vyhledání a sběr dat, zjišťování deskriptivních charakteristik,
3. příprava dat, jejich transformace do vhodné podoby (tato fáze bývá nejpracnější),
4. vlastní analýza, hledání informace v datech, vytváření statistických modelů. Využívají se nejrůznější metody od jednoduchých tabelací

a vizualizací až po sofistikované přístupy jako je genetické programování. Asi nejčastěji používanými metodami však jsou logistická regrese s automatickým výběrem proměnných, rozhodovací stromy a neuronové sítě;

5. vyhodnocení výsledků, aplikace zjištěných poznatků a modelů,
6. zpětnovazební kontrola efektivity výsledků a adekvátnost modelů, využití výsledků v praxi.

4.1.2 Principy optimalizace

Aby bylo jasné, z jakých důvodů je zvolen tento postup a proč je kladen takový důraz na analýzu vstupů, musím trochu předběhnout ve výkladu fází výpočtu a již nyní naznačit, jaké principy budou při optimalizaci používány.

Z analýzy současného vědeckého poznání dané problematiky a z navrhnutého modelu pro řešení definovaného typu úlohy vyplývá, jaké metody a algoritmy lze v tomto případě použít. Řešení bude založeno na principech off-line heuristických algoritmů popsaných v kapitole 1.3.1. I když jsou popsané algoritmy určené pro jednodimensionální BP, jejich principy zůstanou stejné i při rozšíření úlohy na tři dimense.

Hlavní princip tedy spočívá v uspořádání objektů a poté jejich postupném nakládání do kontejneru(ů). Toto se může opakovat vícekrát, vždy s jiným pořadím objektů. Bude tím prozkoumáváno více variant řešení. Počet opakování může být teoreticky dán počtem všech možných permutací uspořádání objektů.

Příklad:

Počet permutací (bez opakovaní) pro lineární řazení vstupujících objektů, jejichž počet n = 10:

$$P(n) = n!$$

$$P(10) = 10! = 3\,628\,800$$

Vzhledem k tomu, že těchto permutací je pro rozsáhlejší úlohy neúměrné množství a každá výpočetní iterace trvá určitou dobu, je prakticky nereálné procházet a zkoumat všechny varianty řešení.

Pro třírozměrný typ úlohy BP navíc nemusí být uspořádání objektů v lineární struktuře, což teoretický počet permutací ještě výrazně zvýší.

Z těchto důvodů musíme vytipovat pouze několik variant (permutací objektů), které budou prozkoumány. V rozumném čase je možné prozkoumat nanejvýš několik tisíc

variant, takže se bude jednat skutečně o velmi malé procento. Na to, jaké varianty uspořádání vybrat, má odpovědět právě vstupní analýza objektů. Tuto analýzu proto považuji za stěžejní a nejdůležitější fázi řešení. Tomu také odpovídá, že je fází z hlediska nároků na výpočetní techniku nejnáročnější.

Vzhledem k tomu, jak je model definované 3D BP úlohy složitý, je vhodné každou variantu prozkoumat ještě dalším způsobem – na jejím základě se pokusit rozvinout a vytvořit jiné, odvozené varianty přípustného řešení. K tomu slouží fáze úpravy získaného řešení.

4.1.3 Princip modularity

Z popisu modelu úlohy je patrné, že takto definovaná úloha je velmi složitá. Pro reálné fungování celého systému je vhodné rozdělit ho do několika funkčně samostatných částí - modulů. S každým modulem by mělo být možné pracovat jako se samostatnou a ucelenou jednotkou. Jejich spolupráci bude zajišťovat nadřazený modul obsahující algoritmizovaný postup popsaný v úvodu kapitoly 4.1.

Modely tedy budou zhruba odpovídat jednotlivým fázím postupu řešení. Další samostatné moduly budou navrženy pro speciální činnosti v rámci optimalizace. Zvenčí se mohou jevit jako jakési „černé skříňky“, ale budou mít jasně definované rozhraní, pomocí kterého spolu budou komunikovat. Modulární architektura je pro rozsáhlé úlohy velmi výhodná. Umožňuje flexibilní vývoj a zdokonalování řešení při zachování hlavních principů modelu úlohy. Sestavená aplikace je tak i při své značné složitosti přehledná a flexibilní[9].

Podrobný popis všech těchto modulů a jejich rozhraní je již nad rámec této práce. Při vývoji softwarové aplikace se s nimi však reálně pracuje.

Snad jako jeden jednoduchý příklad uvedu modul nákladového kalkulátoru, který slouží pro výpočet přepravních nákladů (Krit 2). Vstupní rozhraní modulu představují údaje o druhu dopravy, ujeté vzdálenosti, hmotnosti zásilky, popřípadě další parametry. Výstupním údajem je pak cena za přepravu. Z pohledu optimalizace nás pak vlastně ani nezajímá, jakým způsobem byla výstupní hodnota ze vstupních dat vypočtena. Navíc, časem může být současně používaný modul nevyhovujícím. Pak je celkem snadné, nahradit ho jiným modulem, třeba i od externího dodavatele softwarových řešení, a to ať v podobě programové knihovny (dll) nebo moderní webové služby. Stačí aby bylo implementováno definované rozhraní.

4.2 Analytická fáze

Cílem této fáze je maximálně využít výhod off-line přístupu k řešení BP, kdy je možné si předpřipravit údaje vstupující do úlohy. Úloha analýzy je v podstatě deskripce vstupujících dat a nalezení dominantních struktur nebo vazeb, které jsou

[9] Jednotlivé moduly lze navíc použít jako části dalších úloh a aplikací

v datech na první pohled skryty. Příprava těchto údajů musí být cílená a podložená. K tomu právě slouží analýza vstupujících dat. Pozornost je zaměřena především na objekty, které mají být naloženy a na kontejnery, které jsou pro nakládání k dispozici.

Z obrázku Obrázek 4.1 je patrné, že výsledky analýzy budou mít přímý dopad na další fáze řešení:

- následnou přípravu objektů před optimalizací,
- pro nastavení parametrů algoritmů pro získání výchozího řešení,
- nastavení parametrů algoritmů pro úpravu nebo pro přechod na další řešení.

Vlastnosti vstupních údajů jsou zjišťovány vhodným statistickým šetřením. Statistika nabízí celou řadu teoretických dobře prozkoumaných, zdůvodnitelných a ověřených metod pro analýzu dat.

4.2.1 Určení základních tříd objektů

Vlastní třídění do skupin vzájemně si podobných objektů bude následovat až v další fázi řešení. V analytické části je však vhodné provést alespoň základní otypování objektů podle jejich vlastností (především podle omezujících podmínek) vyplývajících z navrženého modelu.

Budou nás zajímat především toto hierarchické rozdělení objektů:

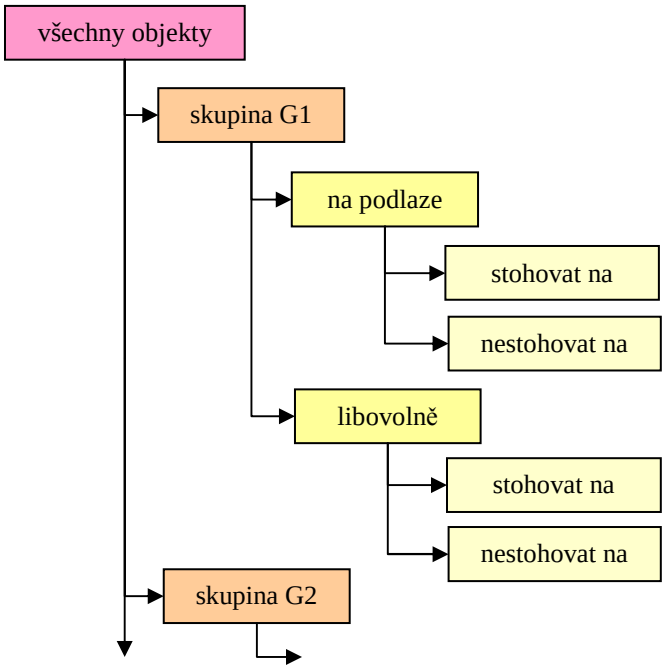
a) podle skupin definovaných omezující podmínkou OP 13 [10],

b) musí být na podlaze,

c) může se na ně stohovat.

S takto zatříděnými objekty (skupinami objektů) lze s výhodou pracovat samostatně. Následná analýza bude provedena pro každou vzniklou základní třídu.

[10] Pokud nejsou tyto omezující podmínky definovány, objekty se chovají, jako by byly v jedné společné skupině



Obrázek 4.2 Strom základních tříd objektů

4.2.2 Analýza nakládaných objektů

Jako nejdůležitější se jeví průzkum množiny nakládaných objektů z hlediska jejich základních parametrů – rozměry, hmotnost a ostatní vlastnosti. Je možné analyzovat celou množinu objektů, ale vhodnější je analyzovat objekty již rozdělené do základních tříd v každé třídě odděleně. Lze sledovat tyto základní deskriptivní charakteristiky:

Četnosti objektů:

- počet objektů ve třídě,
- poměrná četnost objektů vzhledem k celé množině nakládaných objektů,
- míra homogenity (heterogenity) objektů.

Statistické ukazatele pro hmotnost, cenu a rozměry (délka, šířka a výška[11]) a objem objektů:

- minimální a maximální hodnota,
- průměrná hodnota[12],
- modus – nejčetnější hodnota,
- medián – prostřední hodnota v seřazeném souboru,
- směrodatná odchylka,
- variační koeficient.

[11] Rozměry je nutné analyzovat ve všech přípustných polohách naložení objektu, viz 3.2.5.
[12] Slouží také jako pomocná proměnná pro další výpočty.

Míra homogenity se spočítá jako podíl počtu množin stejných objektů ku počtu všem objektů, tedy:

$$\mathrm{H} = \frac{n - n_S + 1}{n} \qquad 4.1$$

H	homogenita
nS	počet množin stejných objektů
n	celkový počet nakládaných objektů

Míra homogenity tedy může nabývat maximální hodnoty 1, když jsou všechny nakládané objekty stejné. Naopak minimální hodnota $\frac{1}{n}$ představuje stav, kdy je každý nakládaný objekt jiný.

4.2.3 Analýza disponibilních kontejnerů

Podobné charakteristiky, jako u nakládaných objektů, lze sledovat u kontejnerů.

Obecné a četnostní ukazatele:

- míra homogenity (heterogenity) kontejnerů,
- četnost uzavřených kontejnerů,
- četnost otevřených kontejnerů.

Opět statistické ukazatele pro hmotnost, cenu a rozměry:

- minimální a maximální hodnota,
- průměrná hodnota,
- modus – nejčetnější hodnota,
- medián – prostřední hodnota v seřazeném souboru,
- směrodatná odchylka,
- variační koeficient.

4.2.4 Analýza omezujících podmínek

Na základě analýzy vlastností vstupních objektů a kontejnerů je vhodné provést výběr potřebných a vytipování nejdůležitějších omezujících podmínek, které budou respektovány v optimalizaci. Na tomto základě bude později vybrán vhodný algoritmus a jeho nastavení.

Velmi důležité je vytipování tzv. dominujícího omezení, dominantní omezující podmínky. Jedná se o parametr, který bude při řešení konkrétní úlohy nejvíce omezující. Například v případě nadměrně těžkých objektů to bude omezující podmínka

dodržení maximální nosnosti. Naopak v případě velkých lehkých objektů to budou prostorová omezení.

Jak jsem uvedl v kapitole 3.2, v modelu rozlišuji několik typů omezujících podmínek, podle jejich působnosti. Podobné hierarchické uspořádání je respektováno i při analýze a testování omezujících podmínek v průběhu řešení.

4.3 Fáze přípravy dat

Na základě úvodního analytického šetření následuje krok přípravy dat pro vlastní optimalizaci. Hlavním cílem je:

- určit typ úlohy,
- uspořádat nakládané objekty do vhodné datové struktury:
  - seskupit objekty do „shluků“,
  - seřadit objekty ve shlucích podle vhodného kritéria,
  - stanovit pravidla mechanismů pro tvorbu variant posloupností vstupních dat,
- uspořádat disponibilní kontejnery,
- na základě typu úlohy a dominantního omezení vybrat vhodné algoritmy (či jejich kombinace) pro řešení úlohy,
- sestavit účelovou funkci na míru typu úlohy,
- připravit sadu parametrů:
  - pro primární optimalizaci,
  - pro úpravu výsledků.

4.3.1 Určení typu úlohy

Hlavním výsledkem analýzy jsou podkladová data pro otypování úlohy podle vstupních dat. Pro konkrétní typ úlohy lze poté použít empiricky nalezené vztahy a pravidla. Na základě těchto pravidel je možno automaticky přizpůsobit parametry optimalizačních metod charakteru nakládaných objektů a tím pro daný typ úlohy dosáhnout kvalitnějšího řešení.

Na určení typu úlohy používám rozhodovací strom. Jeho příprava je demonstrována na zjednodušeném příkladu níže v kapitole 5.1.1. Pro vypěstování rozhodovacího stromu se používá sada trénovacích dat. Každý prvek trénovacích dat představuje úlohu nakládání popsanou pomocí výše uvedených charakteristik. V průběhu práce s algoritmem se budou charakteristiky jednotlivých úloh uchovávat ve znalostní bázi (více v kapitole 4.7) a později je bude možné využít k aktualizaci rozhodovacího stromu. Tím bude zajištěn mechanismus automatického zdokonalování algoritmu pro určení typu úlohy.

4.3.2 Třídění nakládaných objektů

V některých optimalizačních krocích je výhodné pracovat se skupinou vzájemné si podobných objektů. Určit tuto „podobnost“ není pro výpočetní techniku zcela triviální.

Jako výchozí a jednoduché třídění lze považovat základní fixní rozdělení provedené v rámci analýzy (viz kapitola 4.2.1). Ale s tímto tříděním se nespokojíme. Třídění a rozdělení vstupujících objektů má totiž za účel sdružit objekty stejných nebo podobných vlastností do samostatných skupin. Toto roztřídění bude respektováno při uspořádávání objektů v pomocných datových strukturách.

Jednotlivé objekty lze popsat tak, že objekty patřící k témuž konceptu (třídě) mají podobné charakteristiky. Vzhledem k tomu, že jsou objekty popsány hodnotami atributů, lze je teoreticky vzato reprezentovat jako body v mnohorozměrném prostoru, jehož dimenze je dána počtem atributů. Objekty představující příklady jednoho konceptu potom v tomto prostoru tvoří shluky. Cílem je najít vhodný popis těchto shluků a způsoby pro jejich nalezení. Zde lze s výhodou využít osvědčené statistické metody diskriminační a shlukové analýzy.

Diskriminační analýza, neboli úloha klasifikace objektů do předem zadaných tříd, má jistě také své uplatnění. Ze statistického hlediska se jedná o hledání závislosti veličiny určující příslušnost k dané třídě na dalších veličinách.

Podstatnější roli však sehraje shluková analýza[13] (cluster analysis), která odpovídá na otázku, zda lze pozorované příklady rozdělit do skupin - shluků. Je to procedura, pomocí níž objektivně seskupujeme jednotlivé objekty do skupin na základě jejich vzájemné podobnosti a odlišnosti. Vychází tedy z předpokladu, že lze nějakým způsobem poměřovat „vzdálenost“ mezi objekty. Pokud jsou objekty charakterizovány *m* numerickými hodnotami, lze vzdálenost mezi nimi vyjádřit několika různými mírami. Já jsem použil Hammingovu vzdálenost:

$$d_H(x_1, x_2) = \sum_{j=1}^{m} \left| x_{1j} - x_{2j} \right| \qquad 4.2$$

Výpočet jsem při zkoumání ověřoval ještě pomocí Čebyševovy vzdálenosti:

$$d_E(x_1, x_2) = \max_i \left| x_{1j} - x_{2j} \right| \qquad 4.3$$

Dále lze použít například Euklidovskou vzdálenost, tu ale z hlediska vyšší výpočetní složitosti nepoužívám[14]:

$$d_E(x_1, x_2) = \sqrt{\sum_{j=1}^{m} \left(x_{1j} - x_{2j} \right)^2} \qquad 4.4$$

[13] Pro třídění objektů lze použít i jiné metody, například Faktorovou analýzu, metodu optimálního škálování, vícerozměrné škálování, korespondenční analýzu, atd.
[14] Experimentální důkaz o výpočetní složitosti je podán v kapitole 5.1.2

Pro vzdálenosti mezi objekty musí platit tyto podmínky:

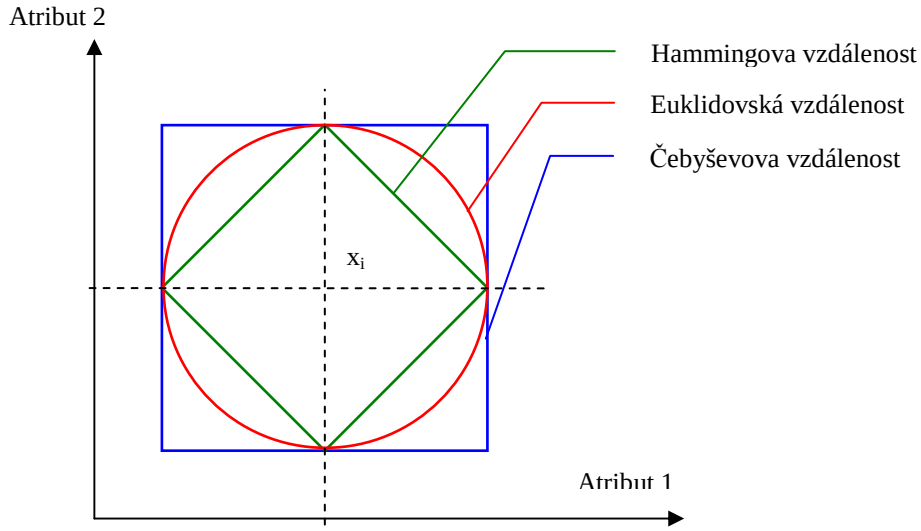
$$d(x_i, x_j) \geq 0$$

$$d(x_i, x_i) = 0$$

$$d(x_i, x_j) = d(x_j, x_i)$$

$$d(x_i, x_j) + d(x_j, x_k) \geq d(x_i, x_k) \qquad 4.5$$

d	vzdálenost mezi objekty
x1, x2	porovnávané objekty
x1i	i-tý atribut objektu



Obrázek 4.3 Vzdálenost mezi objekty měřená dvěmi atributy

Míry vzdálenosti však závisí na měřítku zkoumaných veličin. Proto je nutné veličiny normovat nebo jim dokonce nastavit míru vlivu pomocí váhových koeficientů. Tyto váhové koeficienty jsou jedněmi z parametrů metaalgoritmu, které jsou nastaveny podle typu úlohy a podle dominantního omezení. Taktéž se předpokládá přibližně stejný rozptyl všech veličin.

Pro vlastní analýzu lze použít několik algoritmů. Mě se nejvíce osvědčil upravený algoritmus hierarchického shlukování, kde se začíná v situaci, kdy každý objekt představuje jeden „shluk“. Postupně se pak shluky spojují, teoreticky do té doby než vznikne jediný shluk, který obsahuje všechny objekty. To samozřejmě nemá smysl, proto je potřeba najít vhodnou mez, kdy se shlukováním skončit. Druhá možnost je pamatovat si průběh shlukování a v případě potřeby promítnout shlukování v požadované míře.

Algoritmus lze verbálně popsat těmito kroky:

1. inicializace shluků – každý obsahuje jeden objekt
2. výpočet vzájemných vzdáleností mezi všemi shluky (matice vzdáleností)
3. cyklické operace, dokud není naplněna podmínka konce shlukování

   3.1. spojení dvou nejbližších shluků

   3.2. výpočet a aktualizace vzdáleností od spojeného shluku k ostatním

Vzdálenost mezi shluky lze stanovit několika jednoduchými a z dopravních aplikací důvěrně známými algoritmy:

- algoritmus nejbližšího souseda,
- algoritmus nejvzdálenějšího souseda,
- metodou průměrné vzdálenosti, atd.

Shluky jsou tedy reprezentací objektů podobných vlastností daných podmnožinou atributů.

Příklad:

Úkolem je rozdělit 10 objektů do tří skupin podle jejich rozměrů a hmotnosti. Pro zjednodušení a názornost se předpokládá, že s objekty lze otáčet a lze je klopit. Tabulka 4.1 popisuje tyto objekty a jejich sledované vlastnosti. Poslední sloupec udává příslušnost k výsledné třídě. Toto přiřazení je také zvýrazněno barevně.

objekty	délka	šířka	výška	hmotnost	shluk
1	100	123	76	52	střední
2	122	181	79	26	střední
3	92	71	60	1	malý
4	100	126	62	62	střední
5	156	142	159	41	velký
6	95	110	173	73	velký
7	149	196	53	78	střední
8	140	199	177	37	střední
9	188	139	132	99	velký
10	195	61	155	44	velký

Tabulka 4.1 Příklad shlukování objektů Hammingovou vzdáleností

4.3.3 Řazení objektů

Pro optimalizaci je velmi důležité výchozí seřazení nakládaných objektů. Navržené algoritmy pracují deterministicky, takže výsledek optimalizace záleží kromě parametrů algoritmů především na tomto seřazení.

Ze způsobu seřazení nelze předem poznat, jak bude výsledné řešení kvalitní. Připomínám, že z tohoto důvodu se při řešení prozkoumá několik variant. Jednotlivé varianty se od sebe odlišují (kromě několika proměnných parametrů pro optimalizaci) především tímto řazením a uspořádáním objektů.

Objekty se řadí podle vhodných atributů. Které to budou, záleží na analýze vstupujících objektů a omezujících podmínek. Jedním z výsledků této analýzy je určení dominantní omezující podmínky (nebo více dominantních omezení, pro každou skupinu objektů), viz kapitola 4.2.4.

Nejvhodnější je začít nakládat nejprve ty nejvíce omezující objekty. Pokud je dominantním kapacitním omezením například rozměr v ose X (délka kontejneru), budou objekty primárně seřazeny právě podle své délky. Dále z principů algoritmů platí, že se začíná nakládat od podlahy kontejneru, takže na předních pozicích by měly být objekty, které nemohou být stohovány (musí být na podlaze). Dále budou na přední pozice kandidovat objekty, na které lze stohovat a mají vysokou stohovací hmotnost.

Konkrétně je pro každý nakládaný objekt vypočítán ukazatel „obtížnosti jeho naložení“. Pokud jsou u objektů zadány rozdílné ceny (užitky), je potřeba pro každý objekt ukazatel obtížnosti vynásobit jeho cenou:

$$u_o^i = f\left(Obtížnost_i\right) \cdot \left(1 + k_c \cdot c_i\right) \qquad \forall i \in \textit{množina objektů} \qquad 4.6$$

uo	ukazatel obtížnosti naložení objektu
f(Obtížnost)	funkce pro výpočet obtížnosti
c	cena (užitečnost) objektu
kc	koeficient vlivu ceny objektu na celkovou obtížnost

Objekty jsou poté sestupně seřazeny podle ukazatele obtížnosti.

4.3.4 Tvorba variant, pomocné datové struktury

Jak vyplývá z předchozího popisu, jsou objekty jednak seskupeny podle své podobnosti a déle setříděny podle ukazatele obtížnosti. Sice lze předpokládat, že vzájemně si podobné objekty budou mít přibližně stejnou hodnotu ukazatele obtížnosti, přesto není uspořádání objektů do lineární struktury (prostý seznam) příliš vhodné. Požadována je spíše datová struktura zhruba těchto vlastností:

- počítačově dobře zpracovatelná,
- uspořádatelná,
- snadné a rychlé procházení a vyhledávání objektů ve struktuře,
- flexibilní, snadno modifikovatelná při zachování svých vlastností.

Jako nejvhodnější se nabízí stromové uspořádání dat, která se skládá ze vzájemně propojených uzlů.

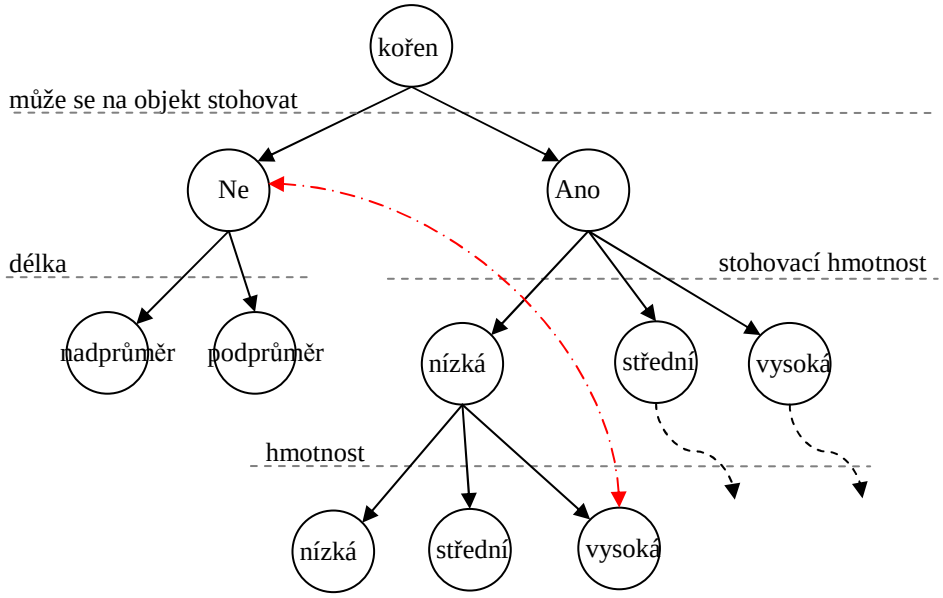
Na tomto místě zavádím pojem „dynamická referenční stromová struktura“ (DRS) pro upravenou stromovou strukturu, která bude svými vlastnostmi vyhovovat stanoveným požadavkům a řešenému problému. Dynamika spočívá v možnosti rychlého přeuspořádání větví stromu a tím vytvoření další varianty „seřazení“ objektů

vstupujících do úlohy. Referenční vlastnost spočívá v dalším druhu vzájemných vazeb mezi některými uzly.

Rozeznávám několik druhů uzlů stromu:

- kategoriální:
  - kořen stromu – vrcholový uzel stromu,
  - vnitřní uzly – představují rozdělení objektů do tříd,
- datové – listové uzly stromu reprezentující jednotlivé objekty.

Uzly jsou navzájem propojeny orientovanými hranami určující hierarchickou polohu uzlu ve stromu. Nejdůležitějším typem uzlu je kategoriální vnitřní uzel. Sdružuje v sobě informace o kategorii objektů v podřízených uzlech a případné referenční odkazy na další uzly. Referenční odkazy existují na základě atributové korelační analýzy a vyjadřují určitý vztah propojených uzlů a samozřejmě podřízených větví stromu.



Obrázek 4.4 Dynamická referenční stromová struktura

4.3.5 Seřazení kontejnerů

Disponibilní kontejnery je výhodné před vlastním výpočtem také seřadit. Řazení bude probíhat pomocí ukazatele „výhodnosti kontejneru“ určeného opět na základě dominantního omezení (kapitola 4.2.4) a ceny kontejneru.

$$u_v^j = f\left(Výhodnost_j\right) \cdot \frac{k_c}{c_j} \qquad \forall j \in množina\ kontejnerů \qquad 4.7$$

uv	ukazatel výhodnosti použití kontejneru
f(Výhodnost)	funkce pro výpočet výhodnosti
c	cena za použití kontejneru

kc	koeficient vlivu ceny na celkovou výhodnost

Kontejnery jsou poté seřazeny sestupně podle této výhodnosti.

4.3.6 Sestava globální účelové funkce

Z analýzy omezujících podmínek vyplývá sada pouze těch omezujících podmínek, které bude třeba respektovat. Ostatní omezující podmínky nebudou při optimalizaci figurovat, čímž lze dosáhnout významného zrychlení výpočtu.

Vzhledem k tomu, že částečně připouštíme porušení některých omezujících podmínek, je na základě analýzy vstupních dat sestaveno hierarchické pořadí vybraných omezujících podmínek podle důležitosti. Každé této podmínce odpovídá nákladová složka vzorce v účelové funkci. Podle důležitosti jsou poté nastaveny váhové koeficienty těchto složek v účelové funkci.

Připomeňme si obecnou podobu účelové funkce, jak byla definovaná v popisu modelu úlohy v kapitole 3.3.5:

$$U = \sum_{k=1}^{pk} x_k \cdot \mu_k \cdot C_k \cdot w_k \qquad 4.8$$

U	hodnota celkové účelové funkce
pk	počet kritérií
x	binární proměnná značící, zda bude kritérium zahrnuto do účelové funkce
μ	znaménková funkce - příznak převodu minimalizačního kritéria na maximalizační (1 nebo -1)
C	cenová hodnota (výsledek) kritéria
w	koeficient, který upravuje váhu (důležitost) kritéria v konkrétním výpočtu

Pro omezující podmínky, které není třeba respektovat bude tedy binární proměnná *x* nastaven na nulu.

Konkrétně se v účelové funkci mohou vyskytnout všechny tyto složky odpovídající kritériím navrženým v kapitole 3.3.1:

- minimalizace ceny použitých kontejnerů,
- minimalizace ceny za přepravu,
- maximalizace užitku naložených objektů,
- maximalizace využité prostorové kapacity kontejnerů,
- maximalizace využití nosnosti kontejnerů,
- minimalizace penalizačních poplatků za nenaložený objekt,
- minimalizace penalizačních poplatků za překročení nosnosti,
- minimalizace penalizačních poplatků za překročení nosnosti,

- minimalizace penalizačních poplatků za nedodržení podélného rozložení hmotnosti,
- minimalizace penalizačních poplatků za nedodržení příčného rozložení hmotnosti,
- minimalizace penalizačních poplatků za překročení rozměrů otevřeného kontejneru.

Pro každou složku je třeba určit koeficienty:

- x zda bude s kritériem počítáno,
- w váhu neboli vliv kritéria.

Koeficienty může zadat ručně přímo uživatel, a to jednorázově pro získání požadovaného řešení, nebo opakovaně při učení algoritmu. Naučený algoritmus poté umí přiřadit výchozí koeficienty pro konkrétní typ úlohy automaticky.

4.3.7 Výběr vhodných algoritmů

Každý typ úlohy má svá specifika a vyžaduje zvláštní postup při svém řešení. V následující kapitole 4.4 bude popsáno několik algoritmů pro získání výchozích řešení.

Algoritmus řešení může zvolit uživatel manuálně, opět buď pro jednorázové řešení konkrétní úlohy, nebo v režimu učení algoritmu. Po naučení lze na základě typu úlohy automaticky stanovit pro konkrétní úlohu vhodné algoritmy a podmínky jejich kombinací.Parametry a pravidla kombinací algoritmů jsou popsány až v následující kapitole.

4.4 Fáze hledání výchozího řešení

Pro získání výchozího řešení používám sadu spolupracujících algoritmů, které jsou však schopny pracovat i samostatně. Ale na základě předchozí analýzy vstupů se může jejich práce účinně kombinovat. Princip modularity tedy platí i zde. Nejprve algoritmy popíši samostatně, poté uvedu možnosti a podmínky jejich spolupráce.

4.4.1 Algoritmus paletizace

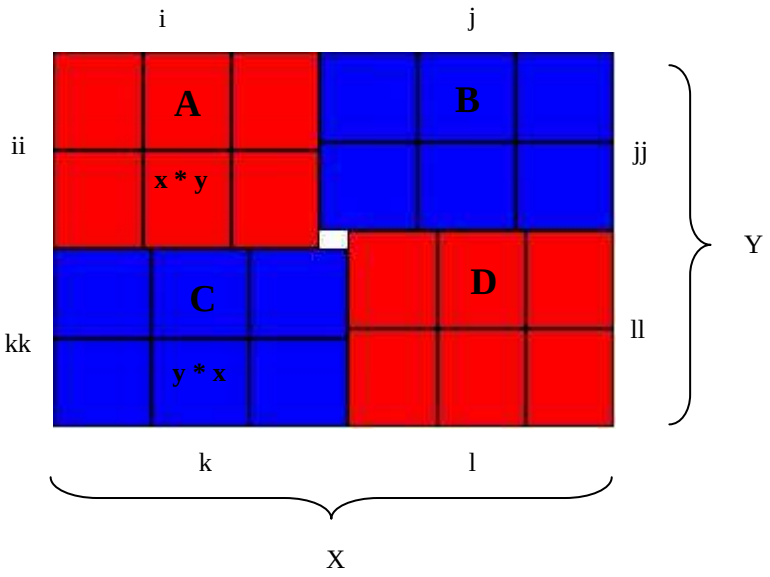
Rychlý, exaktní a poměrně jednoduchý algoritmus pro nakládání homogenní množiny objektů. Využívá právě toho, že nakládané objekty mají stejné vlastnosti, minimálně rozměrové. To je totiž podmínkou k fungování mechanismu, který je schopen prozkoumat všechny varianty.

Obecně lze předpokládat, že nakládaný objekt není omezen co se týče otáčení a klopení. Dále lze odvodit, že objekty naložené ve vrstvě mohou být pouze ve dvou možných polohách. Výška vrstvy bude tedy dána výškou objektu v dané poloze. Výškou vrstvy se tedy nebudeme zabývat a tím dochází k redukci problému na 2D.

Nyní stačí vypočítat parametry vrstvy objektů pro následující 3 varianty dle testovaných rozměrů objektu:

- x, y; výška vrstvy poté bude z,
- x, z; výška vrstvy poté bude y,
- y, z; výška vrstvy poté bude x.

Každá varianta bude naplňovat vzor zobrazený na obrázku Obrázek 4.5, kde je konkrétně popsána první varianta. Vzor se obecně skládá ze čtyř zón: A, B, C a D. Dvě zóny (A a D) jsou pro červené objekty, které jsou naloženy ve své základní poloze „x*y“. Modré zóny (B a C) jsou pro objekty pootočené do polohy „y*x“. Dvojice červených a modrých zón jsou vůči sobě v diagonální poloze.



Obrázek 4.5 Schéma vzoru v algoritmu paletizace

Počet objektů v zónách odpovídá součinu jejich „parametrů“ (počet objektů o daném rozměru v dané poloze), tedy například počet objektů v červené zóně A je:

$$n^A = i \cdot ii \qquad 4.9$$

Z obrázku Obrázek 4.5 je patrná nutnost dodržet rozměrové omezující podmínky:

$$i \cdot x + j \cdot y \le X$$

$$k \cdot y + l \cdot x \le X$$

$$ii \cdot y + kk \cdot x \le Y$$

$$kk \cdot x + ll \cdot y \le Y \qquad 4.10$$

Úkolem algoritmu je nalézt optimální počty stran objektů ve vzoru, aby byla ložná plocha kontejneru maximálně využita, tedy:

$$\min \rightarrow (X \cdot Y) - (x \cdot y \cdot (i \cdot ii + j \cdot jj + k \cdot kk + l \cdot ll)) \qquad 4.11$$

Úlohu lze poměrně snadno vyřešit trojitým iterativním algoritmem, kde budou postupně vyzkoušeny všechny možnosti, například pro počty *i*, *ii* a *jj*. Hodnoty stačí procházet od nuly po maximální možný počet, tedy například:

$$i^{\max} = \frac{X}{x} \text{ (celočíselně)} \qquad 4.12$$

Ostatní parametry lze poté postupně dopočítat. Například z *i* bude vypočteno *j*:

$$j = X - \frac{i \cdot x}{y} \qquad 4.13$$

Výsledkem může být pro každou variantu více stejně kvalitních řešení. V tom případě je vhodné zvolit z nich to nejsymetričtější nebo nejkompaktnější.

Nyní, když jsou připraveny ložné plány pro každou variantu, je potřeba určit četnost vrstev dané varianty, aby byla maximálně využita ložná výška kontejneru. Tento problém lze opět omezit, tentokrát na jednodimensionální BP.

Úkolem je najít četnosti vrstev n_1, n_2 a n_3, aby platilo:

$$\min \rightarrow Z - (nv_1 \cdot z + nv_2 \cdot y + nv_3 \cdot x) \qquad 4.14$$

A to při dodržení omezující podmínky maximální ložné výšky kontejneru:

$$nv_1 \cdot z + nv_2 \cdot y + nv_3 \cdot x \leq Z \qquad 4.15$$

K řešení lze použít jak heuristické algoritmy BP, tak sofistikovanější přístupy lineárního programování. V praxi je však takto malá úloha o počtu 3 neznámých snadno a velmi rychle řešitelná hrubou silou, tedy opět procházením všech variant.

Celkový počet naložených objektů lze nakonec určit vztahem:

$$N = \sum_{v=1}^{3} nv_i \cdot \left(n_i^A + n_i^B + n_i^C + n_i^D\right) \qquad 4.16$$

X	délka ložného prostoru kontejneru
Y	šířka ložného prostoru kontejneru
Z	výška ložného prostoru kontejneru
x	délka nakládaného objektu
y	výška nakládaného objektu
z	šířka nakládaného objektu
i, ii, j, …	počty objektů v dané poloze a umístění (viz obrázek)
A,B,C,D	zóny stejně orientovaných objektů ve vzoru ložného plánu
v	varianta ložného plánu

nA	počet objektů v dané zóně
nv1	počet vrstev dané varianty (1 až 3)
N	celkový počet naložených objektů

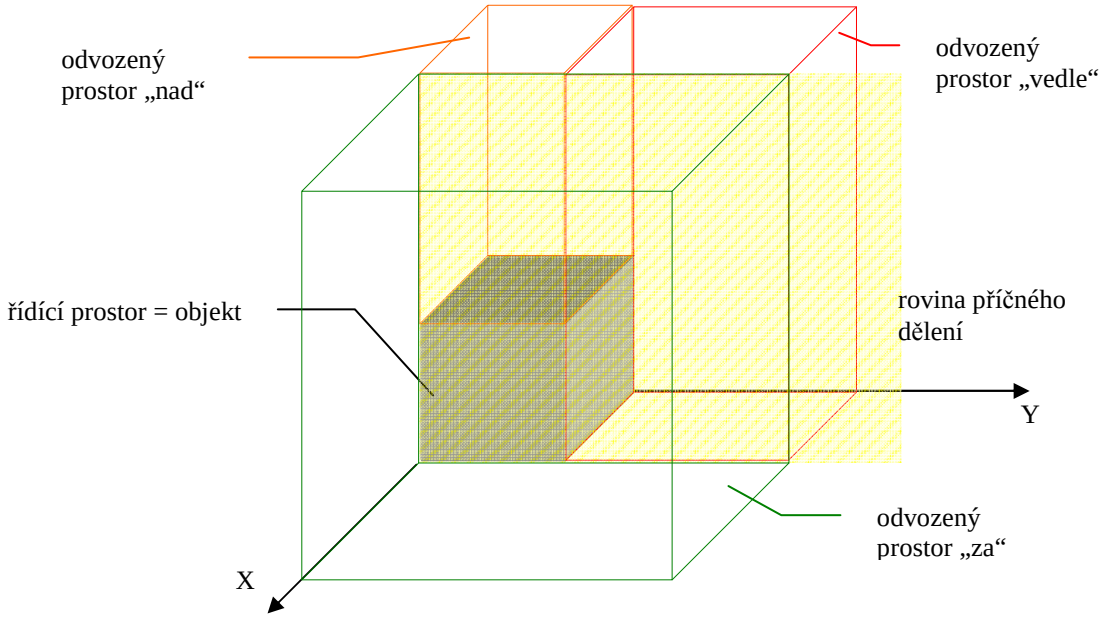
4.4.2 Algoritmus dělení prostoru

Jedná se rekurzivní algoritmus pro nakládání heterogenních objektů implementující heuristický algoritmus Best Fit pro klasický BP problém. Při naložení objektu dojde k dělení 3D prostoru na konvexní podprostory. Do jednoho z nich bude naložen objekt, ostatní budou k dispozici pro další nakládání. Jedná se tedy o aplikaci principu „rozděl a panuj“.

Postup algoritmu lze popsat následujícími kroky:

1. inicializace množiny volných prostorů, množina obsahuje jediný prostor – celý ložný prostor nakládaného kontejneru
2. postupné nakládání objektů podle pořadí
   - 2.1. vyhledání „nejvhodnější“ uspořádané dvojice prostoru *P* z množiny volných prostorů a objektu *O* (nejčastěji se používá kritérium maximalizace využití ložného prostoru)
   - 2.2. pokud je prostor *P* nalezen, realizuje se vlastní naložení objektu
     - 2.2.1. rozdělení prostoru *P* příčným dělením na 4 konvexní podprostory[15] (viz obrázek Obrázek 4.6)
     - 2.2.2. naložení objektu *O* do řídícího podprostoru (svými rozměry odpovídá rozměrům objektu *O*)
     - 2.2.3. odstranění prostoru *P* z množiny volných prostorů
     - 2.2.4. zařazení odvozených podprostorů do množiny volných prostorů
   - 2.3. když vhodný volný prostor neexistuje, tak
     - 2.3.1. pokud je k dispozici jiný volný kontejner pro nakládání, je jeho ložný prostor přidán do množiny volných prostorů a algoritmus se vrací k bodu 2.1.
     - 2.3.2. k dispozici není žádný další kontejner, objekt již nebude možno naložit, je proto z řešení vyloučen
3. konec algoritmu nastává, pokud jsou naloženy všechny objekty nebo je množina volných prostorů prázdná

[15] Podprostory, jejichž rozměry nebudou vyhovovat žádnému objektu (z analýzy) se již tvořit nebudou. Zrovnatak se nebude tvořit podprostor „nad“ objektem, na který nelze stohovat.



Obrázek 4.6 Algoritmus dělení prostorů

Výhody algoritmu jsou následující:

- jednoduchá implementace,
- rychlost (kvadratická, tedy velmi nízká výpočetní složitost),
- algoritmus je díky své obecnosti „volného prostoru“ vhodný pro kombinaci s jinými algoritmy,
- na základě tohoto algoritmu se snadno používají metody na odvození dalších variant řešení (viz kapitola 4.5).

Bohužel výrazněji převažují jeho nevýhody:

- předpokládá seřazení objektů podle velikosti,
- nedosahuje kvalitních výsledků, protože rozdělením prostoru omezuje další možnosti řešení,
- cenou za obecnost je nemožnost flexibilního ověřování omezujících podmínek.

Existuje několik modifikací tohoto algoritmu. Nejjednodušší spočívá ve změně roviny dělení z příčného dělení na dělení podélné. Rovina dělení je pak rovnoběžná s osou X. Žádné nevýhody algoritmu tím eliminovány nejsou, ale změna způsobu dělení prostoru může poskytnout jinou, někdy kvalitnější variantu řešení.

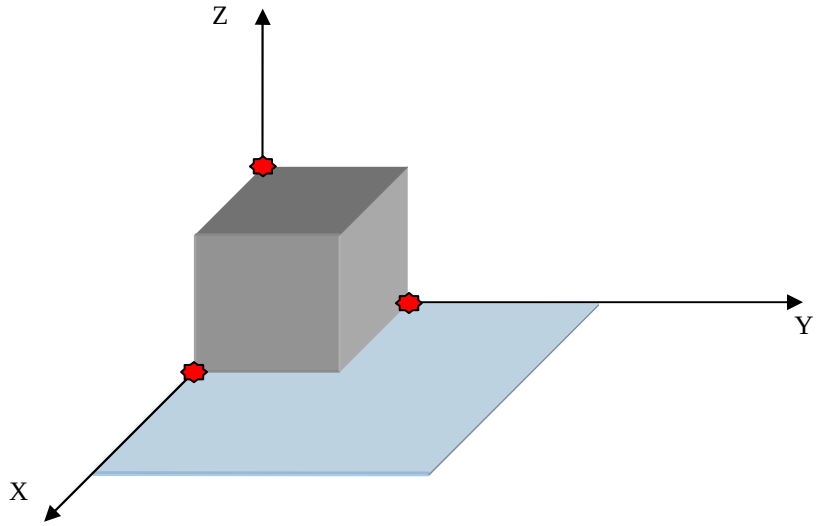
Další modifikace spočívá v obohacení algoritmu o funkce umožňující slučování konvexních podprostorů. Tímto způsobem lze eliminovat nevýhody výchozího algoritmu, ale pouze za cenu zvýšení výpočetní složitosti na exponenciální, tedy zpomalení algoritmu. Prohledávání kombinací prostorů vhodných pro sloučení totiž stojí nezanedbatelný počtní výkon.

4.4.3 Algoritmus vrcholů

Na rozdíl od předchozího algoritmu se místo záznamu celých volných prostorů při nakládání objektů evidují pouze souřadnice polohy podprostorů. Zaznamenávají se tedy jejich „vrcholy“, odtud název algoritmu. Tuto neurčitost musí algoritmus napravit testováním kolize nakládaného objektu s již naloženými objekty.

Algoritmus se skládá z těchto kroků:

1. inicializace množiny vrcholů, obsahuje jediný vrchol o souřadnicích [0;0;0]
2. postupné nakládání objektů podle pořadí
   - 2.1. postupné tvoření uspořádaných dvojic vrcholu *V* a objektu *O*
   - 2.2. otestování přípustnosti přiřazení (naložení) objektu *O* k vrcholu *V*
   - 2.3. pokud existuje přípustné přiřazení
     - 2.3.1. ohodnocení a záznam „výhodnosti“ tohoto přiřazení
     - 2.3.2. nejvýhodnější přiřazení bude realizováno
     - 2.3.3. vrchol *V* bude odstraněn z množiny vrcholů
     - 2.3.4. do množiny budou přidány maximálně 3 nové vrcholy
   - 2.4. když v množině vrcholů nebude žádný vrchol, se kterým by objekt mohl vytvořit přípustné přiřazení, tak
     - 2.4.1. pokud je k dispozici jiný volný kontejner pro nakládání, je jeho „vrchol“ přidán do množiny vrcholů a algoritmus se vrací k bodu 2.1
     - 2.4.2. k dispozici není žádný další kontejner, objekt již nebude možno naložit, je proto z řešení vyloučen
3. konec algoritmu nastává, pokud jsou naloženy všechny objekty nebo neexistuje přípustná uspořádaná dvojice nenaloženého objektu a vrcholu



Obrázek 4.7 Algoritmus vrcholů

Největší problém u tohoto algoritmu je, podle jakého kritéria určit výhodnost onoho přiřazení. Ve výchozím případě je minimalizováno kritérium záboru ohraničeného konvexního prostoru (viz kapitola 3.3.3). Upřednostnění či potlačení postupného nakládání ve směru určité osy je řízeno pomocí dvou trojic parametrů algoritmu:

- pozice naloženého objektu ve směru osy X, Y a Z
- zvětšení ohraničeného konvexního prostor ve směru osy X, Y a Z

4.4.4 Kombinace algoritmů

V kapitole 4.4.2 jsem popsal algoritmus dělení prostoru, který sice nedosahuje příliš kvalitních výsledků, ale má významnou výhodou, že jej využít v kombinaci s ostatními algoritmy. Jde především o to, že naložením jednoho objektu dojde k rozdělení prostoru na několik podprostorů se stejnou množinou charakteristik, ve kterých mohou pracovat libovolné jiné algoritmy.

Jsou dvě hlavní situace, kdy lze kombinace algoritmů s výhodou použít:

1) Existuje poměrně malý počet nadprůměrně velkých objektů (vyplývá z výsledků shlukové analýzy), potom bude vhodné dělit prostor pro pozdější využití podprostorů při úpravách výchozího řešení (viz kapitola 4.5).
2) Homogenita objektů je vysoká (vyplývá z četnostní analýzy objektů), takže lze dynamicky vytvořit „vhodné seskupení“. Tvorba seskupení stejných objektů probíhá pomocí algoritmu paletizace se stupni volnosti, kdy nejsou výsledné rozměry tohoto seskupení pevně určeny.

4.4.5 Parametry algoritmů pro získání výchozího řešení

Po seznámení se s algoritmy pro získání řešení a možnostmi jejich kombinování lze popsat parametry pro získání výchozího řešení.

Jedná se především o parametry, které určují jakým algoritmem bude úloha, případně jednotlivé části úlohy (skupina objektů) řešena:

- typ algoritmu (popřípadě označení kombinací),
- specielní typ pro danou skupinu objektů.

Pro algoritmus dělení prostoru lze použít parametry:

- způsob dělení prostoru, kde 0 = podélné, 1 = příčné, 2 = použití obou způsobů, tedy dvě varianty řešení,
- parametr označující volbu spuštění podpůrného slučování podprostorů.

Pro kombinaci algoritmu vrcholů s algoritmem dělení prostorů je důležitá skupina parametrů pro určení míry dělení prostoru:

- maximální počet primárního dělení v nakládaném kontejneru (počet dělní prostorů, pro které platí, že jejich souřadnice polohy Y = 0),

- koeficient k_{dp} pro dělení prostoru z hlediska osy X. K dělení prostoru dojde, pokud poměr délky nakládaného objektu x ku délce ložného prostoru kontejneru X je větší než tento koeficient.

$$\frac{x}{X} \geq k_{dp} \qquad 4.17$$

Pro dílčí použití algoritmu paletizace jsou určující parametry:

- minimální počet homogenních objektů pro paletizaci,
- maximální velikost (X, Y, Z) výsledného seskupení homogenních objektů[16].

4.5 Fáze úpravy získaného řešení

Varianta řešení získaná v předchozí fázi se vyplatí ještě prozkoumat, popřípadě modifikovat na jinou variantu.

Algoritmy pro získání výchozího řešení nejsou vzhledem k definovanému modelu všemocné. Pokud u nich připustíme možnost, že neověřují splnění všech omezujících podmínek, budou pracovat rychleji a jako řešení poskytnou varianty, ke kterým bychom se jinak nepropracovali. Dočasné přehlížení omezujících podmínek se vztahuje pouze na ty méně významné a řídí se na základě vstupní analýzy pomocí definované sady parametrů. Jsou to omezující podmínky, u nichž připouštíme porušení omezujících podmínek, ale toto je penalizováno v účelové funkci.

Získaná primární varianta řešení tedy nemusí být plně přípustná. Důvodem k odvozování dalších variant je poté snaha o naplnění dočasně vynechaných podmínek omezení. Tato situace se konkrétně týká především omezujících podmínek rozložení hmotnosti.

4.5.1 Možnosti úpravy řešení

Cílem metod na úpravu řešení je tedy přeskupit naložené objekty takovým způsobem, aby splnili ve fázi získání výchozího řešení vynechané omezující podmínky, a tím se snížila hodnota účelové funkce původně navýšená o penalizační poplatky.

Objekty umístěné v kontejneru jsou vhodným způsobem začleněny do tzv. bloků a subbloků. Blokem obecně rozumíme skupinu naložených objektů, kterou lze jasně ohraničit a tím oddělit od ostatních objektů. Subbloky jsou bloky, které svým sloučením naplňují jiný, nadřazený blok.

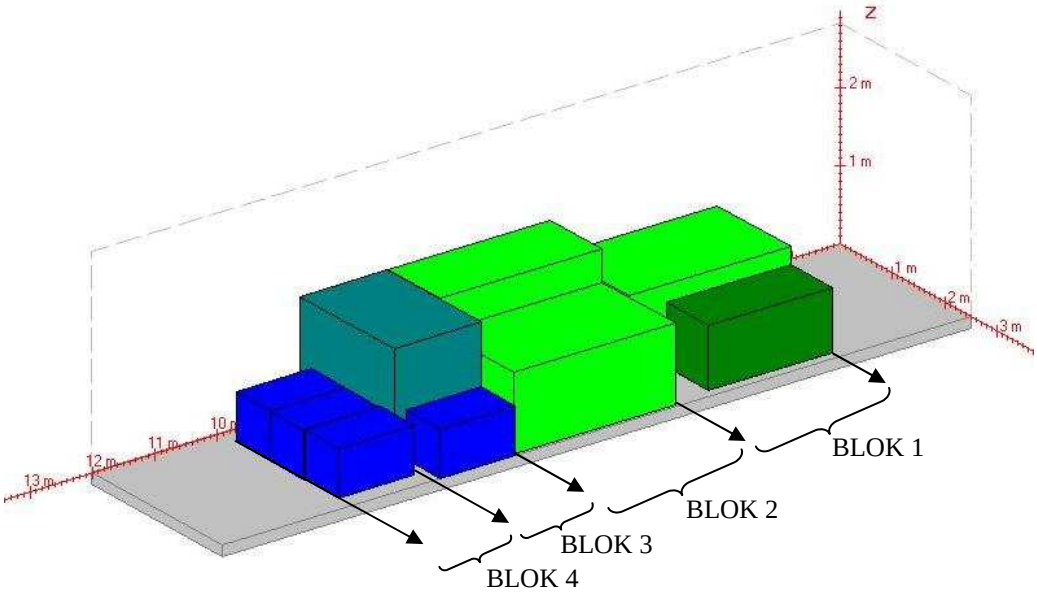
Největším problémem tohoto přístupu ale může být právě určení vhodných bloků. Požadované vlastnosti bloků se budou odvíjet především podle účelu, ke kterému jsou tvořeny, tedy kterou omezující podmínku je třeba řešit a naplnit.

Obecnými požadovanými vlastnostmi bloků jsou:

[16] Pro zjednodušení modelu jsem parametry redukoval pouze na parametr maximální délky seskupení (tedy pouze z hlediska osy X)

- vzájemná podobnost bloků (například co do velikosti a celkové hmotnosti),
- musí existovat nadřazený blok, který zabírá celou šířku ložného prostoru kontejneru.

Ve výjimečných případech lze připustit, že blokem může být i samostatný objekt nebo naopak všechny objekty v kontejneru.



Obrázek 4.8 Bloky naložených objektů

4.5.2 Metoda hledání řezů

Nalezení bloků je hlavním problémem fáze úpravy řešení.

Primární řešení (nebo jeho část) řešené pomocí algoritmu dělení prostorů je velmi vhodné pro následnou úpravu. Bloky totiž při použití příčné řezové roviny dělení vznikají samočinně během optimalizace. Nyní je stačí separovat a zaznamenat do vhodných struktur. Naopak u řešení získaného metodou vrcholů není přítomnost bloků zaručena.

Sestavil jsem proto algoritmus, který bloky v ložném plánu odhalí. Algoritmus využívá vlastnosti, že musí existovat blok, který je přes celou šířku kontejneru. Jeho hlavním úkolem je tedy najít příčné řezy ložným plánem, které by nekolidovaly s žádným objektem.

1. inicializace seznamu objektů v kontejneru
2. seřazení objektů podle jejich umístění $px_i + pl_i^x$ vzhledem k ose X (v obrázku je kritériu řazení vyznačeno červenou šipkou, čísla objektů odpovídají jejich výslednému pořadí)
3. zahájení hledání na začátku ložné plochy kontejneru, případně na konci minulého bloku

4. postupné testování poloh objektů od posledního zkoumaného

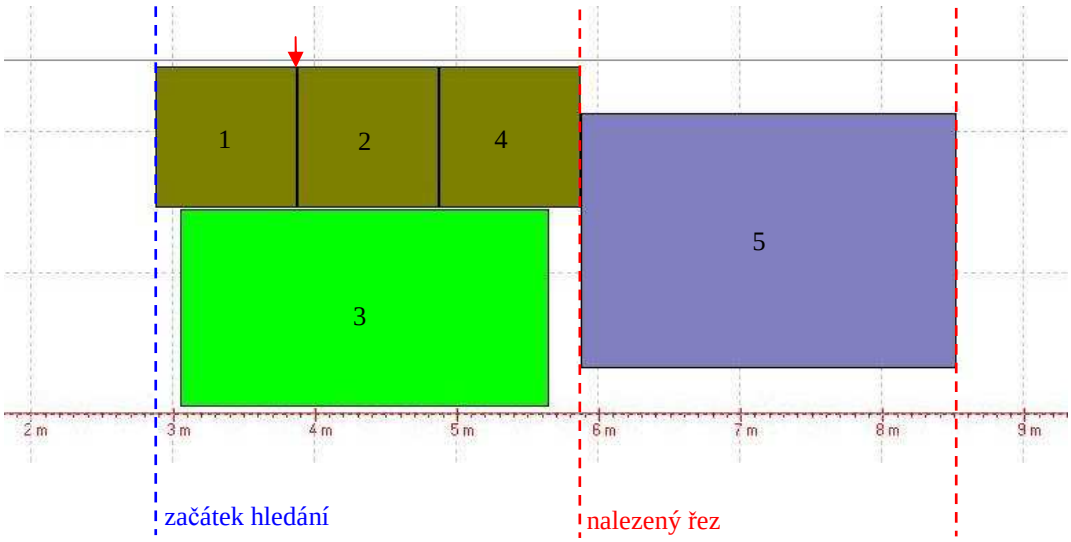
   4.1. porovnání objektu *O* s následujícími objekty *i*

   4.2. když dojde ke splnění podmínky $px_i < px_O + pl_O^x < px_i + pl_i^x$, algoritmu se vrací k bodu 4.1. pro $O = i$

   4.3. pokud ke splnění výše uvedené podmínky nedojde a jsou prozkoumány všechny následující objekty, je nalezen řez v místě $px_O + pl_O^x$, algoritmus pokračuje bodem 3

   4.4. když jsou prozkoumány všechny objekty ($O = n$), je v místě $px_O + pl_O^x$ ukončeno hledání, čímž vznikne poslední blok

px	poloha objektu vzhledem k ose X (vzdálenost od okraje kontejneru)
pl	délka objektu podle jeho orientace



Obrázek 4.9 Výsledky algoritmu hledání řezů (půdorys)

4.5.3 Přechod na jiné řešení

Nové řešení lze získat pomocí vhodné manipulace s celými bloky. Onou manipulací může být jedna nebo více elementárních operací, nejčastěji jejich kombinace.

- Operace mezi kontejnery:
  - přemístění bloku z jednoho kontejneru do jiného,
  - výměna bloků mezi kontejnery.
- Operace uvnitř jednoho kontejneru:
  - změna pořadí bloku v rámci kontejneru,

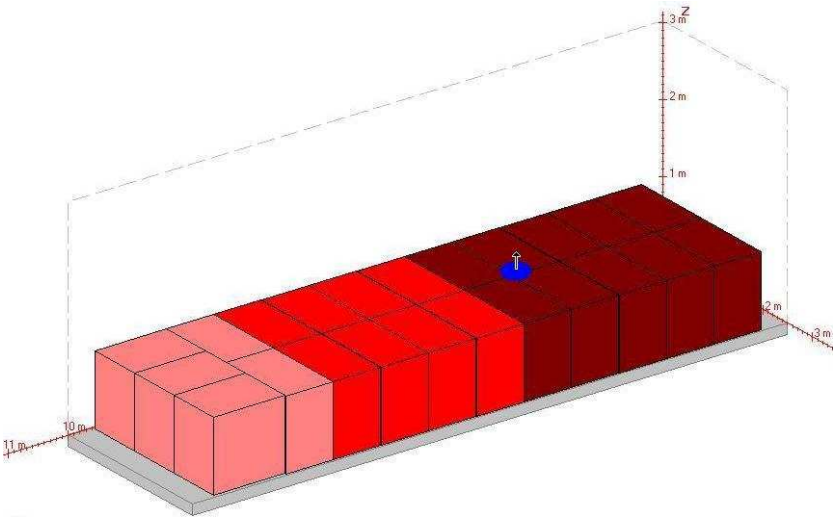
- posun, změna pozice bloku,
- podélná inverze bloku,
- příčná inverze bloku.

Vhodnou kombinací těchto operací lze pak poměrně snadno získat několik dalších variant řešení.

4.5.4 Příklad úpravy řešení

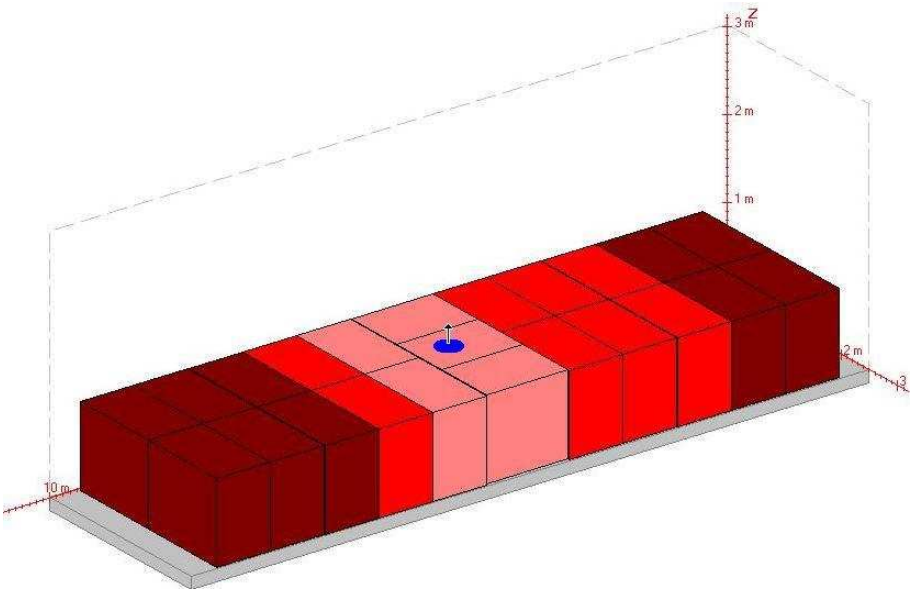
Fungování celého procesu si předveďme na konkrétním případě úpravy řešení pro dodržení omezujících podmínek podélného rozložení hmotnosti a rozložení hmotnosti na nápravy. Zde je úkolem naložit tři druhy palet, které se liší pouze svou hmotností: 50, 400 a 800 kg. Sytost barvy palety odpovídá hmotnosti – čím tmavší barva, tím vyšší hmotnost.

Na prvním obrázku je zachyceno výchozí řešení. Modrá značka představuje polohu těžiště, které je od středu viditelně odchýleno. Poměr zatížení náprav je v tomto případě nepřípustných 2,32.



Obrázek 4.10 Úprava výchozího řešení - nepřípustné naložení

Úpravou této varianty řešení bylo dosaženo řešení mnohem výhodnějšího. Konkrétně byly provedeny operace „Změna pořadí bloku v rámci kontejneru“ a „Podélná inverze bloku“. Poměr zatížení náprav je nyní 1,08.



Obrázek 4.11 Úprava výchozího řešení - opravené naložení

4.6 Prezentace a zhodnocení výsledků

Výsledku úlohy nakládání se říká ložný plán. Lze je prezentovat několika způsoby:

- reprezentativními ukazateli,
- tabelárně,
- graficky.

Nejvýhodnější je kombinace všech způsobů.

4.6.1 Reprezentativní ukazatele ložného plánu

Ložný plán lze pro účely jeho hodnocení popsat pomocí jeho vlastností, nejčastěji ukazatelů poměřující nějakým způsobem jeho kvalitu. Ukazatele mohou být pro celý ložný plán nebo samostatné pro jednotlivé naložené kontejnery.

Ukazatele pro kontejner:

- hodnota účelové funkce pro naložený kontejner,
- procento využití ložného prostoru kontejneru (plocha a objem),
- procento využití nosnosti kontejneru,
- penalizační poplatky za porušení omezujících podmínek naloženého kontejneru.

Celkové ukazatele kvality ložného plánu:

- celková hodnota účelové funkce,
- počet využitých kontejnerů,

- celková suma penalizačních poplatků,
- průměrná hodnota penalizačních poplatků,
- procento nenaloženého počtu objektů,
- průměrné využití ložného prostoru,
- průměrné využití nosnosti.

Tyto reprezentativní ukazatele rozhodně nepopisují způsob naložení objektů. Slouží pouze pro porovnávání jednotlivých ložných plánů, jak pro účely algoritmů, tak pro prvotní hodnocení provedené uživatelem.

4.6.2 Tabelární vyjádření ložného plánu

Ložný plán popsaný tabulkou nebo systémem tabulek je další variantou prezentace výsledků. V tabulce se nachází seznam objektů spolu s parametry jejich umístění v kontejneru, do kterého jsou naloženy.

Tabulka tedy precizně popisuje polohy naložených objektů. Není však příliš vhodná pro hodnocení a porovnávání výsledků.

Naložené kusy: 23

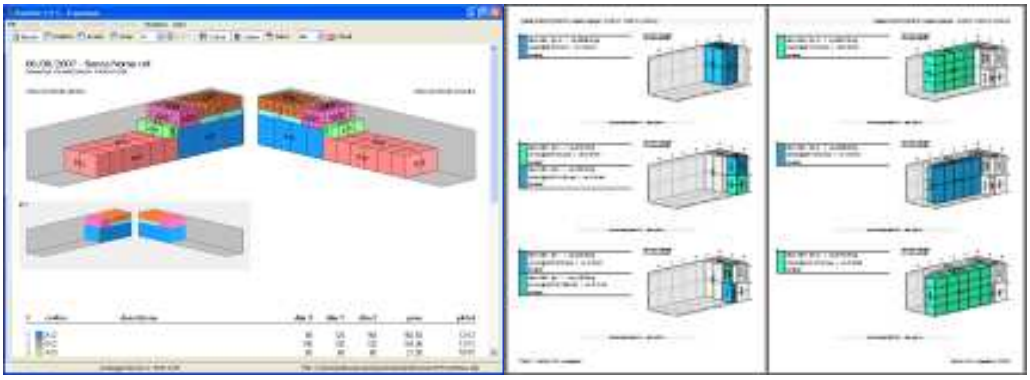
no.	Název	Skupina	Délka	Šířka	Výška	Hmotnos	Poloha	Souř. X	Souř. Y	Souř. Z	Stohov	Otáče	Klope	Mezi	Přes	Na	Sá
1	europaleta 800	0	1200	800	1000	800	2	1200	50	0	2400	X		0	0		
2	europaleta 800	0	1200	800	1000	800	2	1200	1250	0	2400	X		0	0		
3	europaleta 800	0	1200	800	1000	800	2	400	50	0	2400	X		0	0		
4	europaleta 800	0	1200	800	1000	800	2	400	1250	0	2400	X		0	0		
5	europaleta 800	0	1200	800	1000	800	2	7200	50	0	2400	X		0	0		
6	europaleta 800	0	1200	800	1000	800	2	7200	1250	0	2400	X		0	0		
7	europaleta 800	0	1200	800	1000	800	2	8000	50	0	2400	X		0	0		
8	europaleta 800	0	1200	800	1000	800	2	8000	1250	0	2400	X		0	0		
9	europaleta 800	0	1200	800	1000	800	2	8800	50	0	2400	X		0	0		
10	europaleta 800	0	1200	800	1000	800	2	8800	1250	0	2400	X		0	0		
11	europaleta 400	0	1200	800	1000	400	2	2000	50	0	1200	X		0	0		

Obrázek 4.12 Tabelární prezentace výsledků (software JetLoad)

4.6.3 Grafické zobrazení ložného plánu

Grafické znázornění ložného plánu je často použito v rámci celé disertační práce. Uživateli poskytuje velmi dobré podmínky pro to, aby si udělal hrubou představu o umístění a polohách naložených objektů i o kvalitě ložného plánu. Naopak pro další datové zpracování či vyhodnocení není tento způsob prezentace vhodný.

Prostý „obrázek“ ložného plánu je přeci jen mírně omezen. Některé objekty se mohou na obrázku zakrývat. Lze to řešit nabídkou různých pohledů nebo snímkem situace v jisté fázi rozpracovanosti ložného plánu. Kromě třírozměrného pohledu může být ložný plán zobrazen z nárysů, bokorysů nebo půdorysně.



Obrázek 4.13 Ložný plán programu packVol ve formátu PDF (www.packvol.com)

4.6.4 Interaktivní grafické zobrazení

Maximální možnosti a komfort jsou zaručeny v interaktivním třírozměrném grafickém zobrazení pomocí počítače a specializovaného programu. Tento způsob prezentace představuje ideální způsob, protože objekty jsou i v obrázku prezentovány jako vykreslení určitých dat, takže je možné s nimi dále pracovat, datově propojovat například se záznamy v tabulce atd.

Díky těmto vlastnostem může uživatel ložný plán komfortně prohlížet:

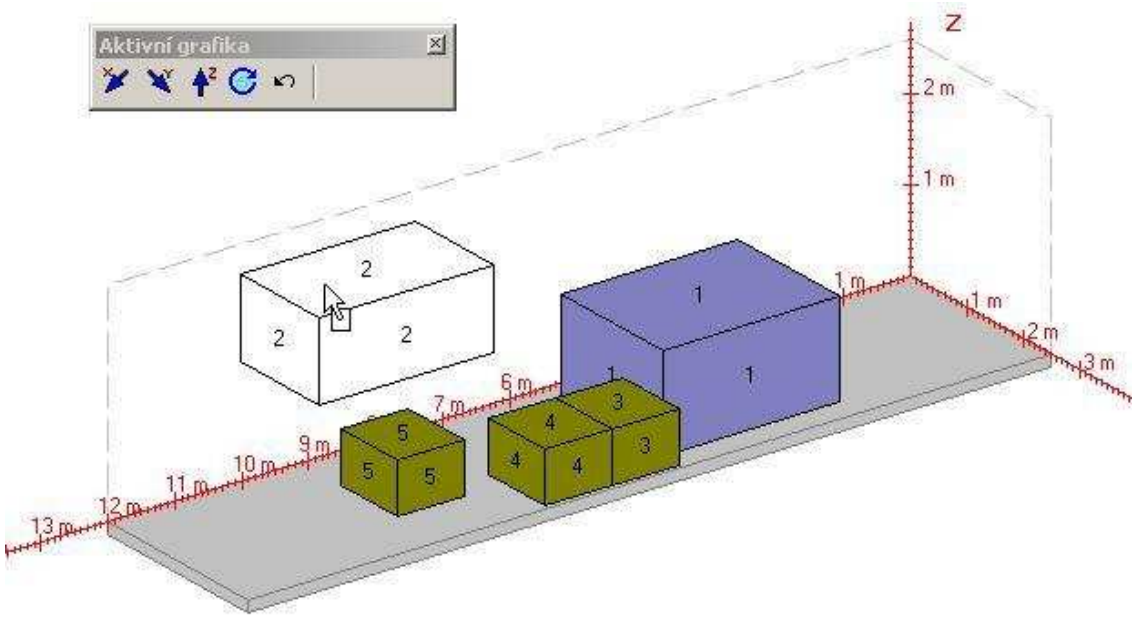
- měnit způsob a úhel pohledu,
- měnit měřítko zobrazení, oddalovat či přibližovat na zobrazení detailů,
- zobrazit těžiště či nerovnoměrnosti naložení nákladu,
- postupně skrývat / zobrazovat jednotlivé objekty,
- měnit způsob vykreslení objektů:
    - plné barevné zobrazení,
    - částečná průhlednost objektů,
    - úplná průhlednost („drátěný model“),
- zobrazovat popisky (čísla, názvy) objektů,
- barevně odlišovat:
    - jednotlivé objekty,
    - skupiny objektů (viz OP 13),
    - bloky objektů (viz kapitola 4.5).

Kromě prohlížení však interaktivní grafické zobrazení nabízí možnost manipulovat s naloženými objekty a tak upravovat řešení:

- změnou umístění objektu (nebo skupiny objektů),
- změnou polohy objektu – lze ho otáčet, případně klopit,

- odstraněním objektu z ložného plánu (vyložení),
- přesunem objektu do jiného kontejneru,
- manipulací s bloky (viz kapitola 4.5):
    - záměna bloků,
    - rotace bloků kolem osy X nebo Y.

Interaktivní grafický systém přitom dokonce umožňuje kontrolovat dodržení rozličných omezujících podmínek.



Obrázek 4.14 Interaktivní grafický systém v aplikaci JetLoad

4.7 Zpětná vazba

V předchozí kapitole bylo uvedeno, jak může uživatel pomocí interaktivního grafického rozhraní manuálně upravit automatické řešení nebo dokonce přímo sestavit nové řešení. Toto manuální řešení je pomocí definovaných ukazatelů kvality porovnáno s automatickým řešením. Pokud je manuální řešení kvalitnější (prokazuje lepší parametry reprezentativních ukazatelů z kapitoly 4.6.1), je na místě, aby se algoritmus z tohoto manuálního řešení „poučil“. Toto učení obecně spočívá v aktualizaci různých parametrů používaných při řešení úlohy a v získávání zkušeností při řešení různých typů úloh. Postupně tak bude docházet ke korigování a upřesňování výchozích znalostí pro řešení úlohy Bin Packingu. V procesu „učení“ je však stále nezbytná aktivní účast lidského faktoru.

Cílem je znát pro úlohu specifickou svými vstupy sadu optimalizačních parametrů, tedy nalézt vztahy mezi vstupy – typem úlohy – parametry optimalizace – řešením

a jeho charakteristikami. Na základě principu analogie, lze pak při řešení úlohy použít řešení, které se již dříve osvědčilo při řešení úloh stejného typu.

4.7.1 Znalostní báze

Znalosti jsou reprezentovány v podobě již vyřešených nakládacích úloh. Tato databáze se také někdy nazývá znalostní báze. Z praktických důvodů je znalostní databáze udržována v relační databázi.

Pro dobré fungování mechanismů učení je potřeba zaznamenávat celou řadu údajů, historických a statistických dat. Jedná se především o záznam celých úloh nakládání včetně jejich řešení, a to v této datové struktuře:

- zadání:
    - seznam objektů k naložení,
    - seznam disponibilních kontejnerů;
- analýza[17]:
    - deskriptivní charakteristiky objektů,
    - deskriptivní charakteristiky kontejnerů,
    - dominující omezení;
- parametry metaalgoritmu:
    - typ úlohy,
    - parametry metaalgoritmu odpovídající typu úlohy;
- řešení:
    - ukazatele kvality řešení[17],
    - podrobný ložný plán.

Pro jednu úlohu může existovat více sad údajů o jejím řešení a k tomu použitých algoritmech.

4.7.2 Aktualizace parametrů metaalgoritmu

Pomocí parametrů popsaných v kapitole 4.2 byla určena vhodná struktura a pořadí objektů vstupujících do úlohy. Podle tohoto pořadí byly objekty postupně nakládány do kontejneru. Dejme tomu, že uživatel nebyl s výsledky automatického řešení spokojen, takže si jej pomocí funkcí interaktivního grafického rozhraní upravil do požadované podoby.

Nyní lze požadovat, aby se algoritmus z uživatelových úprav poučil. Objekty jsou již v kontejneru naloženy a jejich pozice jsou upraveny manuálním zásahem uživatele.

[17] Analytické ukazatele a charakteristiky lze sice znovu vypočítat, ale z hlediska rychlosti výpočtů a vzhledem k tomu, že se nejedná o velký rozsah dat, je nevhodnější je uložit.

Pokud toto řešení dosahuje lepších kvalitativních ukazatelů než původní automatické řešení, potom může být zapojen proces učení[18].

V rámci učení je potřeba vyřešit inverzní úlohu nakládání, zpětně vypočítat nové hodnoty parametrů metaalgoritmu, jejich účinnost ověřit a případně ještě upravit. Tento proces lze charakterizovat tímto postupem:

1) analýza a dekompozice ložného plánu

   1.a) získání pořadí naložených kontejnerů

      1.a.1) zpětné odvození parametrů metaalgoritmu pro získání tohoto pořadí

   1.b) získání výchozí struktury objektů (DRS)

      1.b.1) zpětné odvození, jakými hodnotami parametrů by byl DRS do této podoby sestaven

2) testovací výpočet pomocí nově získaných parametrů na stejné úloze

   2.a) porovnání kvalitativních ukazatelů:

      - původního automatického řešení,
      - manuálního řešení,
      - automatického řešení s novými parametry;

3) testovací výpočet pomocí získaných parametrů na jiných úlohách stejného typu

   3.a) porovnání kvalitativních ukazatelů původních řešení:

      - původního automatického řešení,
      - automatického řešení s novými parametry;

4) při špatných výsledcích řešení s novými parametry jo možné:

   4.a) parametry upravit zkombinováním s hodnotami původních parametrů a vrátit se k testování k bodu 2)

   4.b) nebo prohlásit řešenou úlohu za jiný typ úlohy, který bude nově založen. V každém případě to vede ke změnám parametrů metaalgoritmu, které jsou použity pro určení typu úlohy (viz níže)

5) na základě výsledků testů zamítnutí nebo přijetí získaných parametrů a jejich zápis do znalostní báze

[18] Proces učení může být spuštěn i manuálně, například když se upravené řešení více vyhovuje uživateli, ale z pohledu hodnotících ukazatelů nedosahuje kvalit automatického řešení.

5 Výsledky řešení a jejich analýza

5.1 Výsledky testování algoritmů

5.1.1 Tvorba rozhodovacího stromu pro určení typu úlohy

Výstupem analýzy vstupních dat (kapitola 4.2) je množina atributů, které popisují zadanou úlohu nakládání. Analyzovaná data lze obecně popsat v matici D:

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} \end{bmatrix} \qquad 5.1$$

n	počet objektů
m	počet atributů
x	hodnota atributu

Řádky reprezentují jednotlivé entity vstupních dat (jednotlivé úlohy nakládání), sloupce pak odpovídají atributům. Zde není zatím potřeba rozlišovat, jestli se jedná o atributy kategoriální nebo spojité.

Předpokládejme, že existuje atribut *y*, který popisuje zařazení objektu do některé třídy. Vzhledem k našim záměrů se jedná o tak zvaný „cílový atribut“. Ostatní jsou „vstupní atributy“.

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} & y_1 \\ x_{21} & x_{22} & \cdots & x_{2m} & y_2 \\ \vdots & \vdots & & \vdots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{nm} & y_n \end{bmatrix} \qquad 5.2$$

Objekt s určením třídy, do které náleží lze poté zapsat:

$$o_i = [x_i, y_i] \qquad 5.3$$

Klasifikační úlohu potom můžeme chápat jako úlohu o nalezení takové vhodné funkce *f*, která by hodnotám vstupních algoritmů přiřazovala vhodnou hodnotu atributu cílového.

$$f : x \rightarrow y \qquad 5.4$$

Rozhodovací funkce je zde chápána v širším pojetí. Muže jí být například i neuronová síť nebo rozhodovací strom.

Já k tomuto účelu používám rozhodovacích stromů. Trénovací vstupní data, která se používají k sestavě stromu, se analyzují a postupně rozdělují do menších

a menších podmnožin, tak aby nakonec v těchto podmnožinách převládala vstupní data charakterizující jeden typ úlohy. Použil jsem k tomu jednoduchý rekurzivní algoritmus nazývaný zkráceně TDIDT (top down induction of decision trees) (1):

1. volba některého atributu, podle kterého budou dělena vstupní data (vytvoření uzlu);
2. rozdělení dat na podmnožiny podle hodnot zvoleného atributu (větvení);
3. existuje-li větev, která dostatečně nepopisuje podmnožinu jako samostatnou třídu, opakuj bod 1.

Klíčovým problémem algoritmu je volba onoho vhodného atributu. Tím je ten atribut, který od sebe nejlépe odliší příklady různých tříd. Výhodnou pomůckou jsou charakteristiky popisující míru informace (z teorie informace) především entropie a informační zisk, ale lze využít i chí-kvadrát test nebo Gini index.

Já jsem konkrétně použil entropii a vzdálenost mezi atributem a třídou, tak jak byla definovaná v kapitole 4.3.2. Entropie je veličina, která udává míru neuspořádanosti zkoumaného systému nebo vypovídací hodnotu informace.

$$H(A_v) = -\sum_{t=1}^{T}\left(\frac{n_t}{n}\cdot\log_2\frac{n_t}{n}\right) \qquad 5.5$$

H(Av)	entropie hodnoty v atributu A
T	počet tříd
nt	četnost zastoupení hodnoty v v třídě t
n	rozsah trénovacích dat

Entropii je třeba vypočítat pro každou hodnotu *v*, kterou může atribut nabývat. Z nich lze vypočíst střední hodnotu entropie pro daný atribut jako vážený součet entropií. Váhy představují relativní četnosti kategorií A_v v trénovacích datech. Pro větvení stromu je vybrán ten atribut, jehož střední hodnota entropie je nejnižší.

Příklad:

Pro názornost uvádím velmi zjednodušený příklad, jak se tvoří rozhodovací strom. Aby byl příklad srozumitelný použiji pouze dva jednoduché atributy na množině objektů náležejících do jedné ze tříd vytvořené dle popisu v kapitole 4.2.1. Výpočetní technika není v tomto směru omezena a pracuje s kompletní množinou tříd a disponibilních charakteristik.

Tabulka 5.1: Trénovací data pro tvorbu rozhodovacího stromu

úloha	počet objektů	homogenní objekty	průměrná hmotnost	typ úlohy
1	vysoký	ne	nízká	B
2	vysoký	ano	střední	A
3	střední	ne	střední	C
4	nízký	ne	nízká	C
5	vysoký	ne	nízká	B
6	nízký	ano	vysoká	A

úloha	počet objektů	homogenní objekty	průměrná hmotnost	typ úlohy
7	střední	ne	střední	C
8	střední	ne	vysoká	B
9	vysoký	ne	nízká	B
10	nízký	ano	vysoká	A
11	vysoký	ne	nízká	B
12	střední	ano	střední	A

Tabulka 5.2: Výpočet entropií jednotlivých hodnot atributů

			typ úlohy			jednotlivé entropie		
atribut	hodnota	četnost	A	B	C	A	B	C
počet objektů	vysoký	0,42	0,2	0,8	0	0,46	0,258	0
	střední	0,33	0,25	0,25	0,5	0,5	0,5	0,5
	nízký	0,25	0,67	0	0,33	0,387	0	0,528
homogenní objekty	ano	0,33	1	0	0	0	0	0
	ne	0,67	0	0,63	0,37	0	0,424	0,53
průměrná hmotnost	vysoká	0,25	0,67	0,33	0	0,387	0,528	0
	střední	0,33	0,5	0	0,5	0,5	0	0,5
	nízká	0,42	0	0,8	0,2	0	0,258	0,464

Tabulka 5.3: Výpočet entropie atributů

atribut	entropie atributu
počet objektů	0,985
homogenní objekty	0,639
průměrná hmotnost	0,862

Již z tabulky Tabulka 5.1: Trénovací data pro tvorbu rozhodovacího stromu byla zřejmá souvislost mezi hodnotou atributu „homogenní objekty“ = „ano“ a typem úlohy A. Následné výpočty tuto souvislost potvrzují – atribut „homogenní objekty“ má nejnižší hodnotu entropie (viz tabulka Tabulka 5.3) a proto pro prvotní větvení vybereme tento atribut. Další větvení bude podle atributu „průměrné hmotnosti“ a nakonec až podle „počtu objektů“[19].

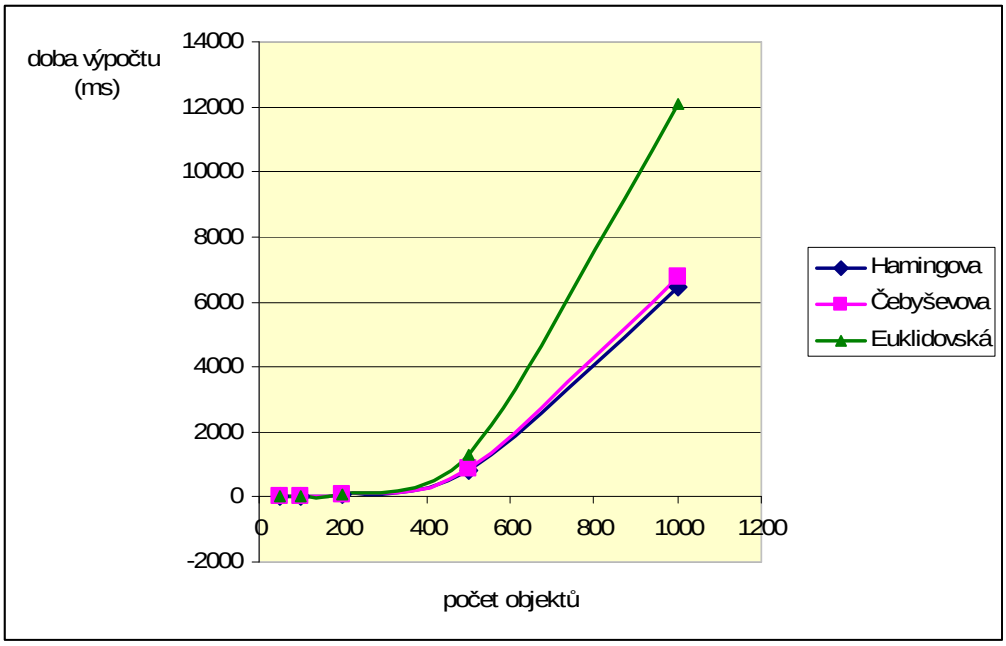
Algoritmus TDIDT dobře funguje pro kategoriální typy atributů, které mohou nabývat relativně nízkého počtu možných hodnot. Počet podmnožin vytvořených v druhém kroku algoritmu odpovídá tomuto počtu hodnot. Problém je se zpracováváním spojitých atributů nebo atributů jejichž obor hodnot je velký, protože nelze pro každou hodnotu vytvořit samostatnou větev. Pomůckou je rozdělení oboru hodnot na intervaly, které lze poté považovat za diskrétní hodnoty atributu. Zde používám principu binarizace atributu, tedy rozdělení na dva intervaly. V tom, kde obor hodnot rozdělit, pomáhá opět entropie.

[19] Počet objektů by neměl nijak výrazně ovlivňovat typ úlohy a následnou volbu metody, což potvrzuje i provedený výpočet.

5.1.2 Test rychlosti algoritmu shlukování při použití různých metrik

Při testování byla použita postupně použita množina 10, 50, 100, 500 a 1000 náhodně vygenerovaných objektů. Toto testovací shlukování se provádělo pouze podle atributů vyjadřujících rozměry objektů. Shlukování bylo zastaveno, až když byly objekty roztříděny do 3 skupin – „malé“, „střední“ a „velké“ objekty. Zjištěné časy jsou v milisekundách.

počet objektů	50	100	200	500	1000
Hamingova vzd.	4	19	66	815	6473
Čebyševova vzd.	3	20	72	860	6748
Euklidovská vzd.	5	23	98	1309	12076



5.1.3 Určení parametrů algoritmu vrcholů

Připomeňme, že výhodnost umístění objektu do daného místa (vrcholu) v kontejneru je řízeno sadou dvou trojic parametrů:

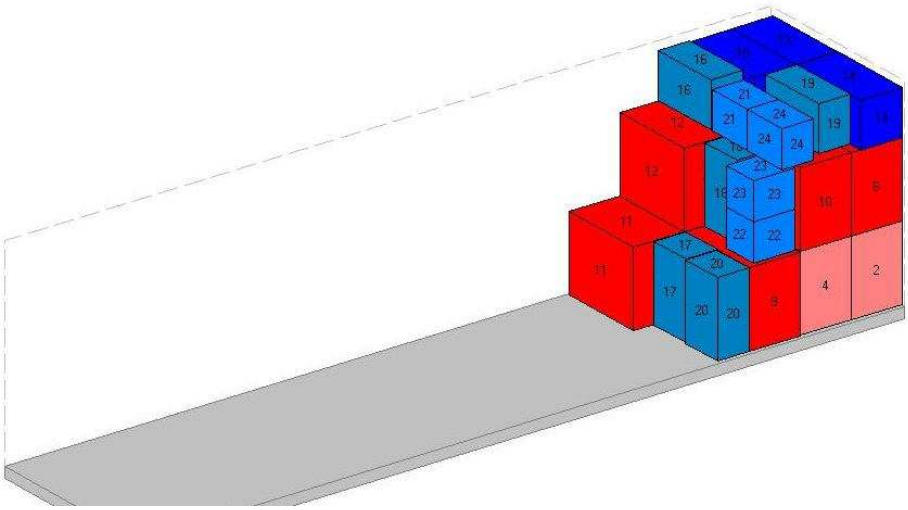
- pozice naloženého objektu ve směru osy X, Y a Z,
- zvětšení ohraničeného konvexního prostor ve směru osy X, Y a Z.

Nastavení těchto parametrů značně ovlivňuje způsob naložení.

Pro nakládání k čelu kontejneru budou tyto parametry nabývat například hodnot:

[1, 10, 100], [100, 1000, 10000]

Výsledek naložení je poté patrný z obrázku Obrázek 5.1:

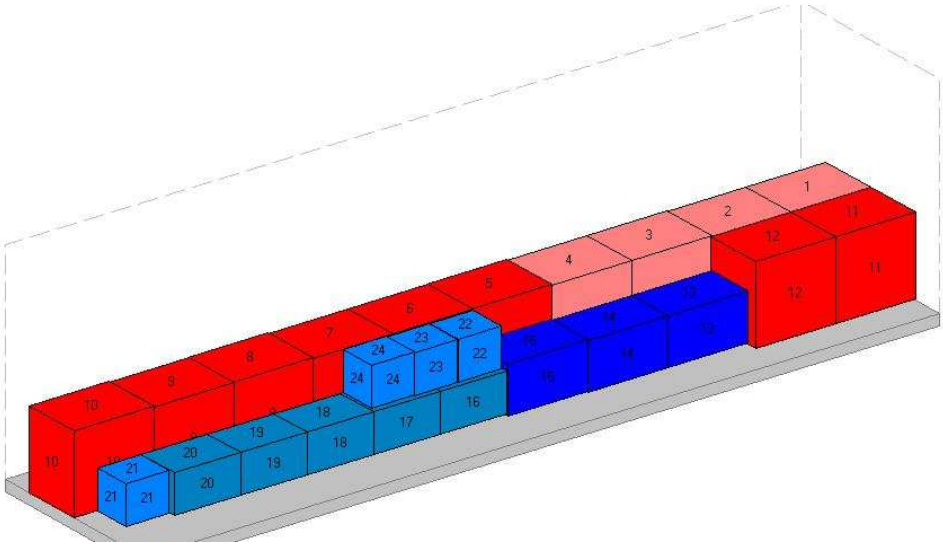


Obrázek 5.1 Výsledek nakládání – algoritmus vrcholů 1

Pro nakládání na podlahu kontejneru budou parametry nabývat hodnot:

[100, 10, 1], [10000, 1000, 100]

Výsledek a rozdíl oproti předchozí variantě je opět zřejmý z obrázku Obrázek 5.2:

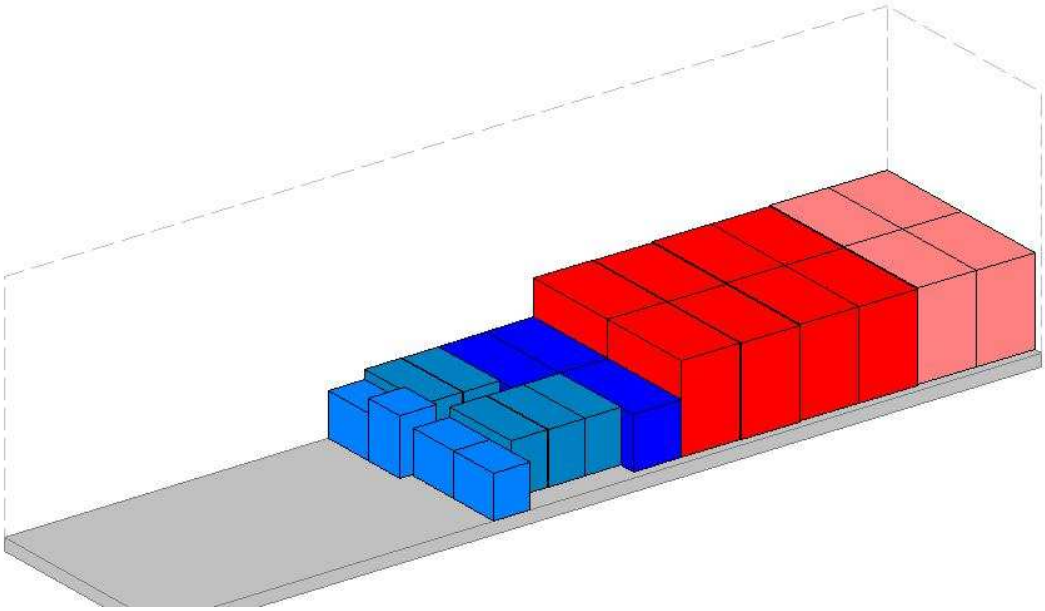


Obrázek 5.2 Výsledek nakládání – algoritmus vrcholů 2

Porovnáním obou extrémních variant, je zřejmé, že výsledky jsou naprosto rozdílné. Ani jeden výsledek nemusí být v praxi přijatelný[20]. Proto dejme učícímu se algoritmu za úkol nastavit parametry algoritmu vrcholů tak, aby naložené objekty byly ve stabilních polohách (tedy na podleze) a aby bylo naložení vhodné pro další automatické úpravy (tvorba loků objektů – viz kapitola 4.5). Po zpracování vhodných trénovacích dat nastavil algoritmus parametry následovně:

[20] Při výpočtech byly vynechány hmotnostní omezující podmínky, aby byl zřejmý vliv parametrů na výsledek nakládání.

[162, 7, 1391], [1576, 203, 2093],
čemuž odpovídá ložný plán zobrazený na obrázku Obrázek 5.3



Obrázek 5.3 Výsledek nakládání – algoritmus vrcholů 3

5.2 Proces učení a správa znalostí

5.2.1 Aktualizace znalostí

Výsledky získané rutinním provozováním softwarové aplikace je potřebné shromažďovat a analyzovat a poté zjištěnými pravidly a znalostmi doplňovat a aktualizovat parametry algoritmů. V navrhované softwarové aplikaci je to možné dvojím způsobem:

- Lokální aktualizace znalostí spočívá ve specializaci znalostí algoritmu používaného jedním uživatelem na řešení množiny konkrétních úloh typických pro tohoto uživatele. Hlavním znakem je tedy specializace.
- Centrální aktualizace znalostí spočívá v tvorbě nových typů úloh a zobecňování znalostí a jim odpovídajících pravidel. Vzniká na základě analýzy velkého množství řešených úloh od většího počtu uživatelů.

Představme si, že aplikace řešící problém optimalizace nakládky běží u několika nezávislých uživatelů. Při dodávce obsahovala aplikace jen základní znalostní bázi pro řešení klasných úloh tohoto problému. Jednotliví uživatelé jsou propojeni s nějakou centrální aplikací a databází pro správu znalostí. Tito uživatelé pak mohou posílat popisy řešených úloh se získaným či upraveným řešením do centrální databáze. Tam se v určitých časových intervalech podle výše uvedených principů učení provede analýza a otipování úloh a následná syntéza potřebných parametrů. Tyto zobecněné parametry metaalgoritmů pro jednotlivé typy úloh mohou být poté k dispozici uživatelům.

5.2.2 Zapomínání

Při dlouhodobém používání systému se počet zaznamenaných úloh stále zvětšuje. Navíc typ obvyklých úloh řešených jedním uživatelem se může časem změnit. Nebo naopak může dojít k přeučení systému, takže ten se bude potýkat s detailními odchylkami obvykle řešeného problému, ale bude mít těžkosti při řešení jiného, z hlediska znalostní báze neobvyklému typu problému. Při automatizovaném učení by se však brala v úvahu celá množina dat, včetně těch neaktuálních. Z těchto důvodů bývá potřebné implementovat také principy „zapomínání“. V oblasti strojového učení se tomu říká „inkrementální učení a zapomínání“.

Nejedná se o nic jiného, než že se pro získávání znalostí („trénování“ algoritmů) nepracuje s celou množinou disponibilních dat, ale jen s určitou jejich relevantní podmnožinou. Záleží na situaci, jak je tato podmnožina definovaná. Nejčastěji se pracuje jen se sadou aktuálních dat, starší data se tedy do učení nezapojují. Dále lze například vynechat data jen určitého konceptu, tedy „zapomenout“ pravidla pro řešení konkrétního typu úlohy. Tento přístup je vhodný v situaci, že došlo k přeučení algoritmu na tento typ úlohy.

5.3 Ukázka kompletního řešení

5.3.1 Popis úlohy

Pro ukázku kompletní činnosti jsem zvolil poměrně jednoduchou úlohu. Náhodně jsem vybral 10 různých typů objektů. Jejich počet k naložení jsem opět určil náhodně tak aby celkový počet nakládaných objektů byl 100. Objekty nelze otáčet ani klopit. Ceny všech objektů jsou nulové. Kontejner jsem zvolil jediný takových rozměrů, aby se do něho všechny objekty nevešly. To proto, aby v tomto příkladě byly více parné rozdíly mezi variantami naložení.

Tabulka 5.4 Příklad nakládaných objektů

Název	Počet	Délka	Šířka	Výška	Hmotnost
kvádr 11	2	625	450	300	5
paleta 8	13	850	1200	599	75
kvádr 2	12	900	1250	599	30
kvádr 13	5	1000	1150	600	121
paleta 2	16	1000	500	600	50
paleta 6	20	1200	800	1200	200
paleta 14	10	1350	850	1700	1600
kvádr 5	11	1500	700	450	20
kvádr 7	9	1850	1050	600	550
paleta 7	2	2000	1000	1200	1000

Tabulka 5.5 Příklad disponibilních kontejnerů

Název	Počet	Délka	Šířka	Výška	Nosnost
MAN 26.463 BDF	1	6900	2450	2400	15000

5.3.2 Analýza vstupů

Kromě statistických veličin uvedených v tabulce Tabulka 5.6 vyplvají z analýzy tyto základní poznatky:

- existuje jediná základní skupina objektů (dle definice v kapitole 4.2.1),
- celkový počet objektů k naložení je tedy 100,
- homogenita nakládaných objektů podle vztahu 3.1 uvedeném v kapitole 4.2.2 je 0,91, tedy 91%,
- dominantním omezením je objem kontejneru, nejkritičtější omezení je v souřadnicích Y (šířka), poté Z (výška) a nakonec X (délka).

Tabulka 5.6 Příklad analýzy vstupujících objektů

	Délka	Šířka	Výška	Hmotnost
minimum	625	450	300	5
maximum	2000	1250	1700	1600
průměr	1227,5	895	784,8	365,1
směrodatná odchylka	423,15	268,75	412,43	509,28

5.3.3 Příprava dat a řešení

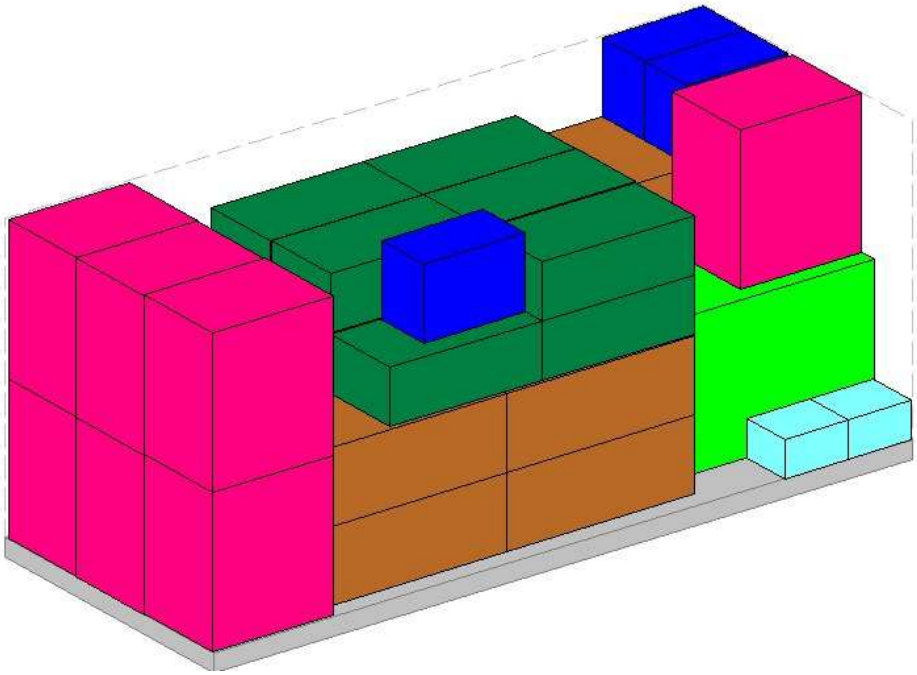
Vzhledem k dominantnímu omezení budou objekty hierarchicky setříděné podle svých rozměrů, a to šířky, výšky a nakonec délky.

V první variantě řešení vynechám vliv dříve získaný znalostí pro řešení, abych simuloval inicializační výpočet bez určení typu úlohy a následného nastavení parametrů metaalgoritmu. Výsledek naložení je zaznamenán na obrázku Obrázek 5.4 Příklad naložení 1, některé ukazatele ložného plánu pak v tabulce Tabulka 5.7 Porovnání ukazatelů naložení.

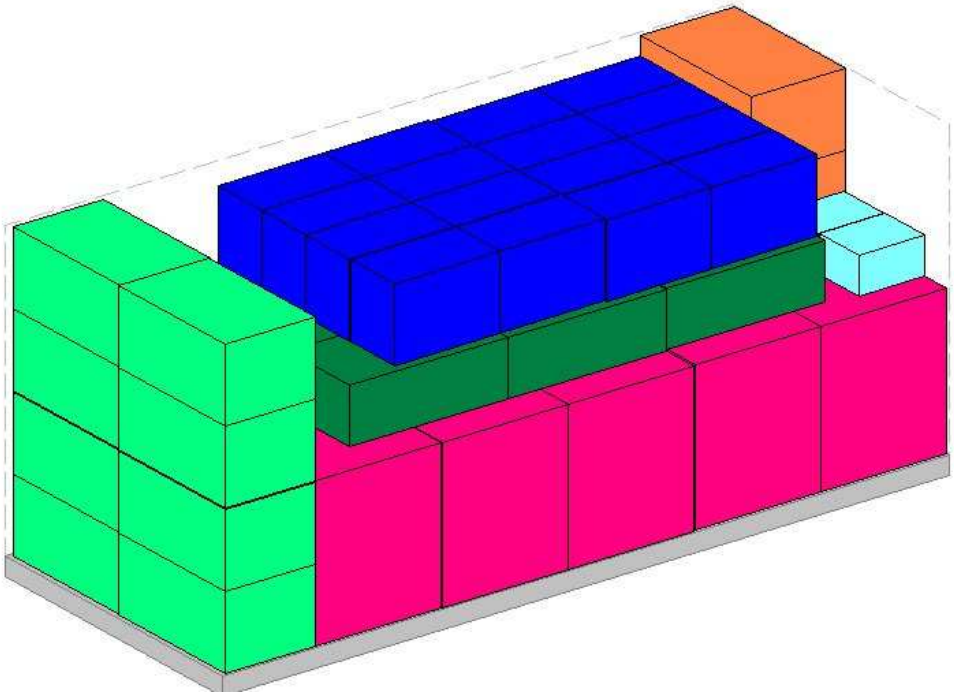
Další varianta řešení tohoto příkladu byla získána již se zapojením znalostní báze - byl určen typ úlohy, pro který již existuje sada parametrů, které se použily při výpočtu. Výsledek je opět popsán pomocí ukazatelů v tabulce Tabulka 5.7 Porovnání ukazatelů naložení a zobrazen na obrázku Obrázek 5.5.

Tabulka 5.7 Porovnání ukazatelů naložení

varianta	počet objektů	zatížení	využití nosnosti	poměr kol	poměr náprav	využití plochy	vyžití objemu
1	34	8730	58,2 %	1,17	1,14	89,9 %	73,0 %
2	52	4650	31,0 %	1,08	1,03	97,3 %	80,7 %



Obrázek 5.4 Příklad naložení 1



Obrázek 5.5 Příklad naložení 2

Z tabulky Tabulka 5.7 Porovnání ukazatelů naložení i souvisejících obrázků je zřejmé, že druhá varianta naložení je obecně výhodnější. Celková naložená hmotnost, a tedy využití nosnosti kontejneru, je však téměř poloviční. To je způsobeno tím, že se při druhém řešení algoritmus ještě více zaměřil na využití dominantní omezující podmínky využití objemu, ale naopak nebral v potaz hmotnost objektů (v tomto případě méně významnou).

6 Přínosy disertační práce

6.1 Přínosy pro další rozvoj vědního oboru

V disertační práci je detailně zpracován model úlohy 3D Bin Packingu:

- přehled a rozdělení omezujících podmínek, jejich praktické použití a vliv na řešení,
- přehled možných kritérií a jejich matematické vyjádření,
- popis datových struktur,
- popis požadovaných výstupů.

Hlavním přínosem disertační práce je však rozšíření metodického aparátu pro řešení úloh tří-dimensionálního Bin Packingu:

- návrh specializovaných datových struktur,
- speciálně vyvinuté algoritmy pro optimalizaci nakládky,
    - algoritmus paletizace,
    - algoritmus dělení prostoru,
    - algoritmus vrcholů,
- nově navržený metaalgoritmus,
    - obecný postup řešení,
    - důraz na analytickou fázi a na přípravu vstupů do řešení,
    - unikátní propojení a možnosti kombinování jednotlivých algoritmů optimalizace,
    - implementace principů umělé inteligence a strojového učení,
- další podpůrné algoritmy,
    - metoda hledání řezů,
    - algoritmus pro automatizované učení, atd.

6.2 Praktické přínosy

Model řešené úlohy Bin Packingu byl sestavován především s ohledem na možnosti jeho praktického využití. Zahrnuje nejdůležitější praktické požadavky a omezení, uvažuje praktická kritéria. Model úlohy i jeho vlastní zpracování pomocí navržených metod a algoritmů jsou navíc díky modulární architektuře otevřené pro případné další rozšíření a praktické použití.

Výsledkem, již nyní pro praxi částečně využitelným, jsou také vyvinuté softwarové aplikace.

6.3 Softwarové aplikace

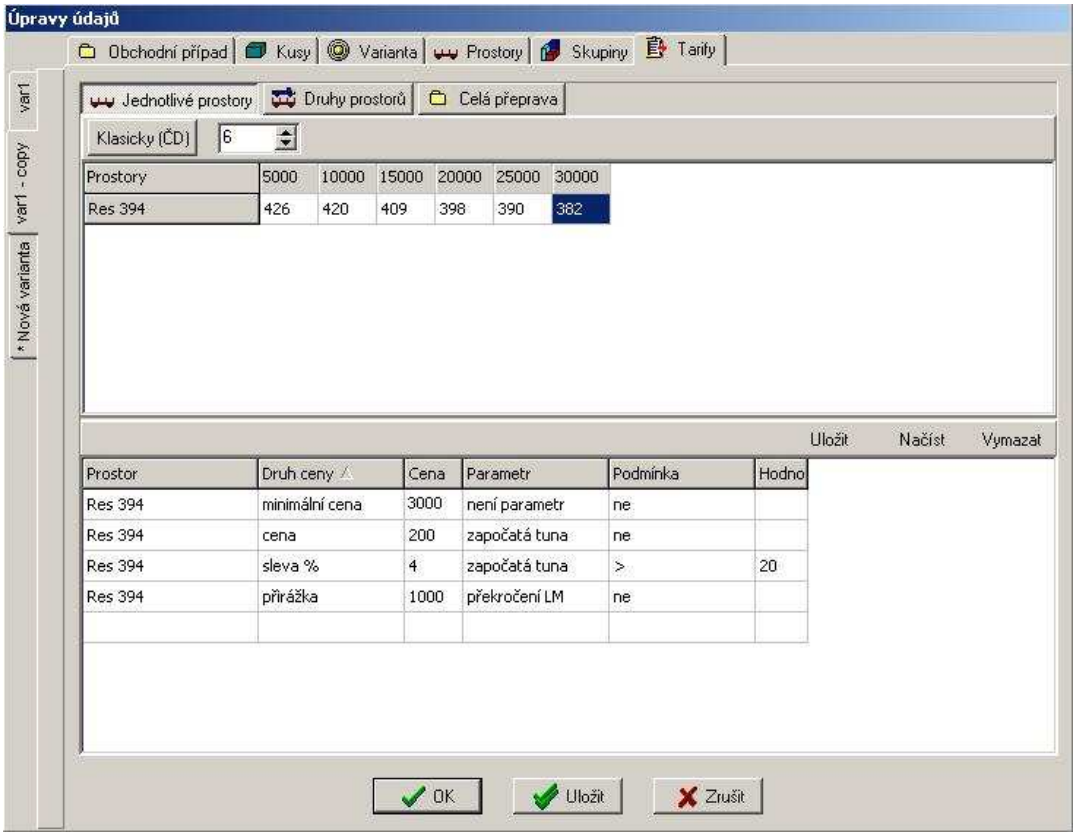
Model, popsaný v kapitole 2, jsem implementoval v rámci softwarové aplikace s názvem JetLoad. Program rovněž obsahuje většinu v disertační práci popisovaných algoritmů a metod pro optimalizaci nakládky. Některé analytické části a obsluhu znalostní báze, jakožto i procesy učení jsou naprogramovány v několika dalších programových knihovnách. Při zpracovávání problematiky nakládání jsem vytvořil pro účely testování i některé další samostatné programy.

6.3.1 JetLaod

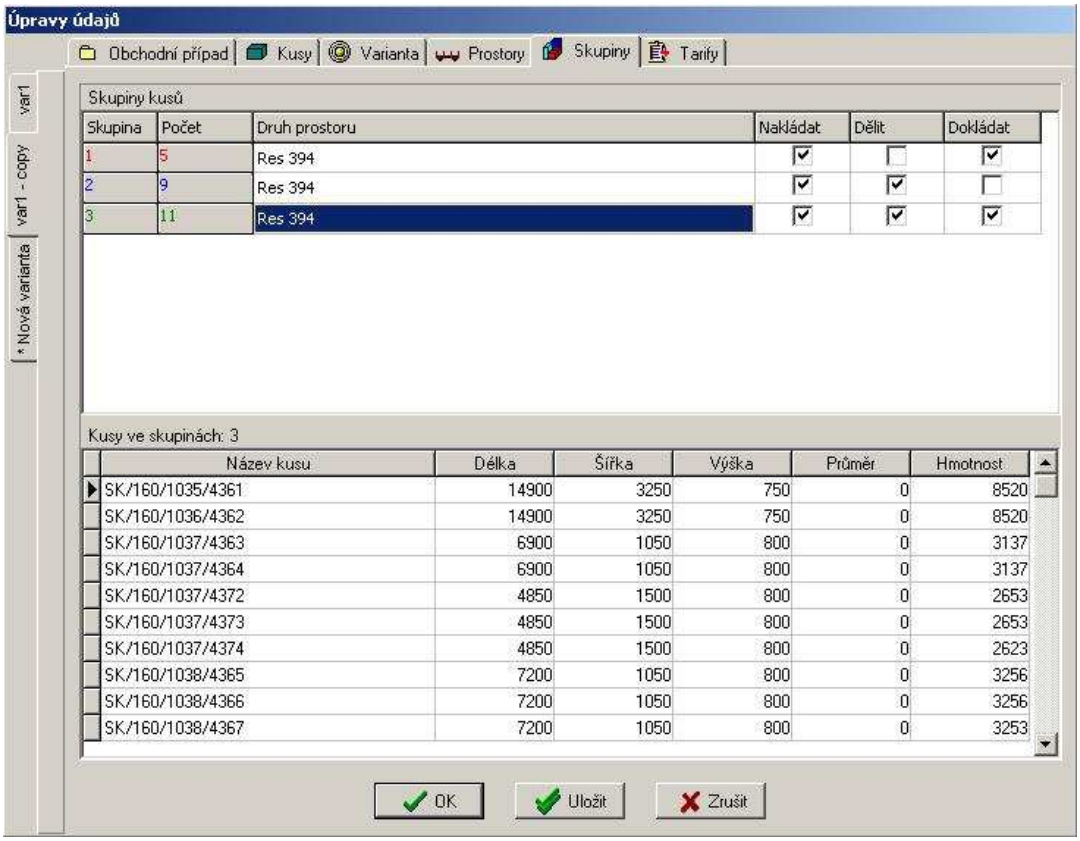
JetLoad je program, který samočinně navrhne naložení určeného zboží na vybrané dopravní prostředky. Prvotním cílem je minimalizace ceny za přepravu, dalším pak dodržení technických parametrů dopravních prostředků vzhledem k jejich vlastnostem i vlastnostem dopravní cesty. Naprogramován je v programovacím jazyku Borland Pascal, prostředí Delphi. Databáze je typu MS Access.

Program splňuje většinu vytyčených praktických požadavků:

- obsahuje kompletní model zahrnující celou řadu v praxi důležitých omezujících podmínek,
- jako hlavní kritérium uvažuje celkovou cenu za přemístění, tomu odpovídá i cenotvorný modul (viz obrázek Obrázek 6.1),
- umožňuje nakládat zboží ve skupinách (viz obrázek Obrázek 6.2 Zadání skupin zboří a jejich vlastností v programu JetLoad),
- obsahuje interaktivní grafický systém popsaný v kapitole 4.6.4,
- umožňuje „uzamknout“ částečné řešení a provést optimalizaci nakládky zbývajícího zboží,
- umožňuje hierarchické nakládání – například zboží na palety, palety do kontejnerů, kontejnery na vagóny.



Obrázek 6.1 Modul pro cenotvorbu a zadávání tarifů v aplikaci JetLoad



Obrázek 6.2 Zadání skupin zboří a jejich vlastností v programu JetLoad

6.3.2 Zajišťování zboží a program FixLoad

Z praktického hlediska je důležité zajištění naloženého objektu v kontejneru proti jeho pohybu. Obecně lze použít tyto způsoby zajištění[21]:

- tvarové,
- třecí (přivázání),
- úhlopříčné (přivázání),
- svazkování.

Tvarové jištění spočívá v „opření“ objektu například o čelo kontejneru nebo o jiné naložené objekty. Pokud má být objekt opřen o čelo kontejneru, musí se s tím předem počítat a objekt vhodně umístit v dynamické referenční stromové struktuře (viz kapitola 4.3.4).

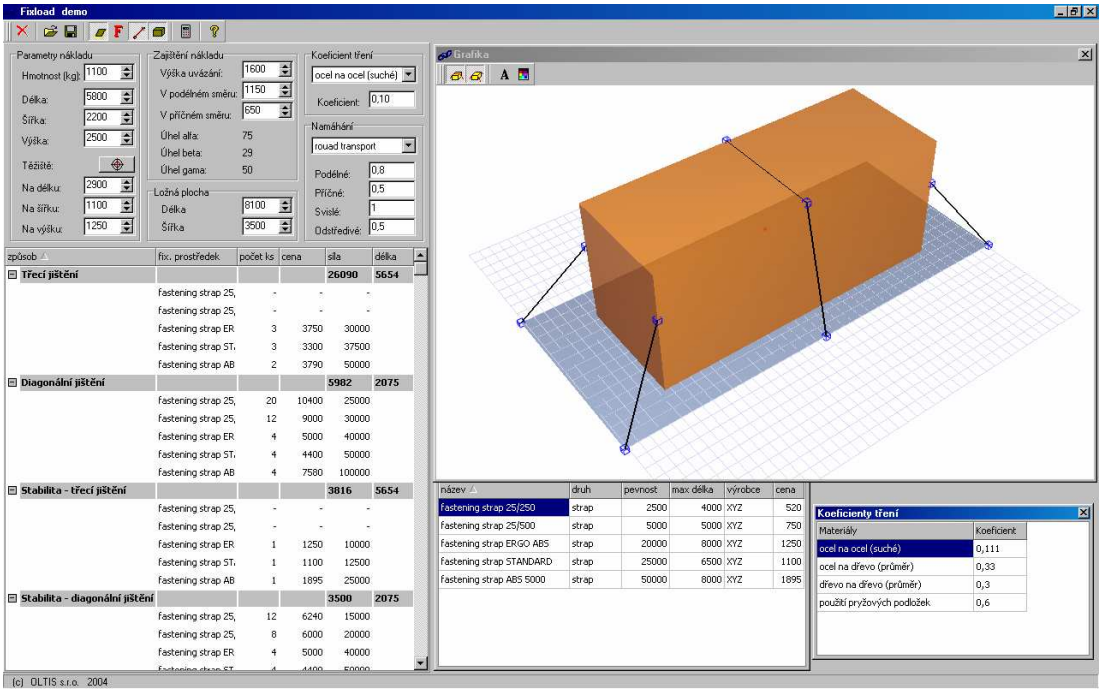
V praxi se rovněž používá způsob zajištění pomocí dřevěných trámců přibitých k podlaze. Zde je tedy podmínkou umístění objektu na podleze kontejneru a druh kontejneru, který toto umožňuje. Se zajištěním pomocí dřevěných trámců lze předem počítat a během optimalizace rozšířit zábor prostoru o šířku dřevěných trámců. Pokud stačí objekt zajistit oporou ostatních naložených objektů, je třeba zohlednit vlastnosti těchto ostatních objektů, aby je případné silové působení nepoškodilo. To však nelze zabezpečit před vlastním průběhem optimalizace, proto se to musí kontrolovat (a výsledek omezovat) až v průběhu optimalizace.

Svazkování spočívá ve spojení často menších homogenních objektů v jeden celek, jehož vlastnosti již lépe odolávají dynamickým vlivům. Použít svazkování se přímo nabízí pro objekty naložené algoritmem paletizace (viz algoritmus popsaný v kapitole 4.4.1).

Třecí způsob jištění spočívá ve zvýšení účinků třecí síly zvýšením přítlaku objektu k podlaze kontejneru pomocí vhodných fixačních prostředků. Přivázání rovněž pomocí fixačních prostředků působí silou proti směru možného pohybu objektu. Týká se tedy objektů umístěných na podlaze kontejneru. Pokud se s tímto jištěním počítá již při nakládce daného objektu lze nakládací prostor omezit o rezervy potřebné k umístění fixačních prostředků.

Třecí a úhlopříčné jištění zboží je řešeno právě v programu FixLoad.

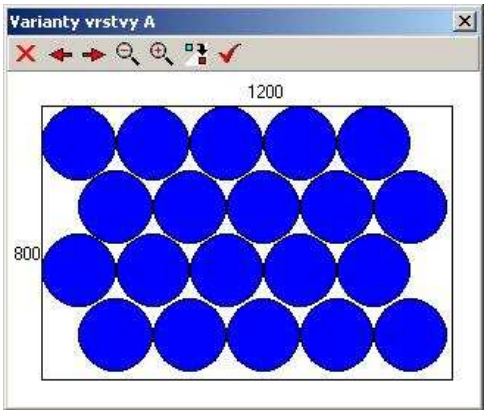
[21] V praxi se často používá kombinace různých způsobů jištění a celá řada dalších fixačních pomůcek.



Obrázek 6.3 FixLoad

6.3.3 Palllaod

Samostatný program implementující pouze algoritmus paletizace, který je popsán v kapitole 4.4.1. Program obsahuje také algoritmus pro nakládání homogenních objektů tvaru válce a umožňuje zásah uživatele do tvořeného ložného plánu.



Obrázek 6.4 Pallload – nakládání homogenních objektů tvaru válce

Grafické uživatelské prostředí je zachyceno také na obrázcích Obrázek 3.4 a Obrázek 4.5, které byly uvedeny v průběhu předchozího textu.

6.3.4 Moduly umělé inteligence

Vzhledem k principům modularity je většina metod pro analýzu dat a algoritmů učení implementována v programových knihovnách (dynamicky linkované knihovny DLL), které nemají grafické rozhraní. Proto je bohužel není možné dokumentovat

obrazovými přílohami. Jejich činnost je však podrobně popsána v kapitole 4.7 a ukázkové příklady nastiňující jejich principy jsou v kapitolách 5.1 a 5.2.

Pro řešení dílčích problémů pomocí neuronových sítí jsem použil volně dostupné knihovny funkcí AForge.NET. Jedná se o framework pro vývojové prostředí C#, který je určený pro vývojáře a výzkumné pracovníky v oblasti počítačového vidění a umělé inteligence. Pokrývá tyto základní oblasti:

- neuronové sítě,
- genetické a evoluční algoritmy,
- strojové učení,
- zpracování obrazu,
- počítačové vidění - detekce pohybu.

6.4 Možnosti dalšího výzkumu

I když je navržený model poměrně rozsáhlý a komplexní, existují vždy další možnosti zkoumání dané problematiky. Detailní pohled na určitou část problému dává sice nějaké odpovědi, ale zároveň vyvolává řadu dalších otázek. V této kapitole bych proto rád přednesl některé úvahy nad dalším možným vývojem řešení této problematiky, a to především v oblastech:

- hlubšího provázání algoritmů pro optimalizaci nakládky s algoritmy počítající působení dynamických vlivů na zásilku během přepravy,
- učení a získávání znalostí z výsledků řešení nakládacích úloh,
- analyzování získaných znalostí.

6.4.1 Získávání znalostí

Část vize, jak by mohl celý systém fungovat v praxi je již naznačena v kapitole 5.2.1. Jedná se jen o teoretickou úvahu, jak získávat velký počet rozmanitých a hlavně praktických úloh nakládání. Bohužel nebylo možné tento způsob získávání dat prakticky ověřit. Je to zapříčiněno především nutností zapojit do projektu další subjekty, dopravce či spediční společnosti. Pro to by bylo také nutno vytvořit profesionální a hlavně komplexní softwarovou aplikaci, což je velmi pracné a nákladné. Nemluvě o ochotě zmíněných subjektů spolupracovat a poskytovat svá soukromá data.

6.4.2 Analýza získaných znalostí

Pro další rozvoj algoritmů umělé inteligence by bylo vhodné najít vztahy mezi vlastnostmi objektů a parametry optimalizace. Cílem by bylo nalezení zdrojů závislosti

v nominálních datech, tedy nalezení obecných asociačních pravidel, která popisují především:

- vztahy mezi kombinacemi hodnot binárních atributů,
- korelace mezi numerickými atributy podmíněné kombinací kategoriálních atributů.

To by posloužilo především ke generalizaci některých získaných znalostí, tedy vytvoření obecně platných pravidel.

Hledání zajímavých souvislostí mezi hodnotami různých atributů je možné pomocí souboru metod explorační a konfirmační analýzy. Cílem explorační analýzy je vyčíst z dat maximum informací a zajímavých souvislostí, a to vzhledem k nějakému obecnému problému. Tím se explorační analýza liší od konfirmační analýzy dat, která se zabývá testováním předem přesně formulovaných hypotéz.

Pro hledání korelací mezi vlastnostmi objektů a parametry optimalizace či jinými parametry lze použít celou řadu konkrétních statistických metod:

- Korelační analýza se používá k zjištění, zda mezi dvěmi veličinami existuje nějaká lineární závislost. Jednoduchost, přehlednost a názornost při aplikaci ji činní vhodnou pro prvotní rychlou analýzu.
- Analýza rozptylu umožňuje posoudit rozdíly mezi průměry různých veličin. Pro víc než jeden zkoumaný znak je to naopak metoda značně výpočetně náročná.
- Kontingenční tabulky se ve spojení nejčastěji s chí-kvadrát testem používají pro zjišťování vztahu mezi dvěmi veličinami. Chí-kvadrát test je založen na vyhodnocení rozdílu mezi četnostmi jednotlivých kombinací vlastností uvedených v kontingenční tabulce a četnostmi očekávanými při platnosti testované hypotézy o nezávislosti daných veličin.
- Regresní analýza slouží pro zjišťování funkční závislosti jedné numerické veličiny na jiných numerických veličinách. Na rozdíl od korelační analýzy, která zkoumá, zda závislost existuje, nás zde zajímají přímo parametry této závislosti. V nejjednodušším případě hledáme parametry lineární funkce, které můžeme získat pomocí metody nejmenších čtverců. V našem případě by měla větší význam mnohorozměrná regrese. Lineární mnohorozměrná regrese zkoumá lineární závislost jedné veličiny na více nezávislých veličinách. V případě nelineární regrese se předpokládá složitější funkční závislost mezi veličinami, například polynomickou, exponenciální nebo logistickou.

ZÁVĚR

Problémy typu Bin Packing se v praxi vyskytují poměrně často. Díky tomu se jedná o velice rozsáhlou a různorodou oblast. Jedno však mají všechny typy úloh BP společné – výpočetní složitost. Neexistuje totiž postup, kterým lze získat exaktní řešení rozsáhlejších úloh v reálném čase. Proto byla vyvinuta celá řada heuristických algoritmů pro řešení úloh Bin Packingu. Některé z nich jsou obecněji použitelné, ale pouze za cenu nepřesných výsledků. Naopak, jiné algoritmy jsou poměrně úspěšné, ale pouze při řešení úzce specifikovaných typů úloh.

Cílem disertační práce bylo rozšířit metodický aparát pro řešení úloh Bin Packingu a otevřít nové cesty k řešení tohoto problému. V práci je navržen zcela nový přístup z hlediska řešení úloh Bin Packingu. Tento obecný postup umožňuje dosáhnout kvalitních řešení i při značném rozšíření modelu úlohy o praktické požadavky a omezující podmínky. Počítá sice s interaktivními zásahy uživatele, ale implementuje také některé principy strojového učení.

Díky tomuto postupu lze „sestavit“ vhodný algoritmus pro řešení konkrétního typu úlohy až v průběhu jejího řešení. Těžištěm navrženého postupu řešení je statistická analýza do úlohy vstupujících údajů. Na základě jejích výsledků a „zkušeností“ získaných z předešlých výpočtů a uložených ve znalostní bázi je na základě vytvořených algoritmů optimalizován ložný plán.

Celý tento postup je v disertační práci podrobně popsán a ilustrován řadou příkladů a obrázků. Jeho funkčnost byla prakticky ověřena pomocí vytvořených softwarových aplikací.

RESUMÉ

Disertační práce se zabývá problematikou nakládání a optimalizace využití ložného prostoru. Důraz je kladen především na praktickou úlohu nakládání. Při optimalizaci se využívá rychlých heuristických metod, interaktivního přístupu a principů strojového učení.

V první části je daná problematika analyzována. Jsou popsána teoretická východiska problému, jeho členění a praktické využití. Dále jsou uvedeny některé základní modely a algoritmy používané v současnosti k řešení, je analyzována jejich výpočetní složitost a posouzena praktická použitelnost.

Druhá kapitola podrobně popisuje model daného problému, datovou strukturu, omezující podmínky a kritéria.

Třetí kapitola předkládá zcela nový postup řešení. V něm je kladen důraz především na analýzu vstupních dat. Součástí je i popis navržených algoritmů a principů strojového učení.

Čtvrtá závěrečná kapitola prezentuje dosažené výsledky. Na několika příkladech jsou ukázány principy získávání řešení a jejich hodnocení. Jsou zde také představeny vyvinuté softwarové aplikace, které byly při řešení použity.

Klíčová slova:

nakládání, optimalizace, výpočetní složitost, analýza, heuristické metody, Bin Packing, strojové učení, rozhodovací stromy

THE ABSTRACT

This written report deals with loading problems and optimization of usage of loading space. Practical loading task is studied especially. Fast heuristic methods, interactive approach and machine learning principles are for optimization used.

The first part analyses this problem, describes the theoretical basis of this issue, its structuring and practical applications. Several basic models and algorithms for solving Bin Packing Problem are introduced here. The computing complexity and possibilities of practical usage of algorithms are analysed in this part.

The second part describes model of this loading problem in details. Binding conditions and criterions are emphasising here.

The third part brings a new common method of solution. There is analyse of incoming data accentuate above all. Description of proposed algorithms and machine learning principles are there too.

The last part describes assumptions and benefits for solutions of Bin Packing Problem in this thesis. Purchase of solutions and their evaluation are shown here. Software used for this solving is introduced.

Keywords:

loading, optimization, computing complexity, analysis, heuristic methods, Bin Packing, machine learning, decision trees

POUŽITÉ INFORMAČNÍ ZDROJE

1) Berka, P., *Dobývání znalostí z databází*, Praha: Academia, 2003, 366 s., ISBN 80-200-1062-9
2) Coffman E.G., Garey M.R., Johnson D.S, *Approximation Algorithms for Bin Packing,* PWS Publishing, Boston 1997
3) Corcoran, A. L., Wainright, R. L., *A genetic algorithm for packing in three dimensions*, Symposium on Applied Computing. Proceedings of the 1992 ACM/SIGAPP symposium on Applied Computing: technological challenges of the 1990's, 1992
4) Data Mining, on-line http://datamining.xf.cz/
5) Faroe O., Pisinger D., Zachariasen, M., *Guided local search for the three-dimensional bin packing problem*. Technical Report 99-13, Department of Computer Science, University of Copenhagen, 1999, on-line http://citeseer.nj.nec.com/faroe99guided.html
6) Hoffman K., *Set Covering, Packing and Partitioning Problems*, George Mason University, New York, on-line http://iris.gmu.edu/~khoffman/papers/set_covering.html
7) Johnson D.S., *Near-Optimal Bin Packing Algorithms*, PhD thesis, Massachusetts Institute of Technology, Department of Mathematics, Cambridge 1973
8) Koziol J., *Optimisation of Usage of Storage Space*, Third Scientific Conference 2003, Pardubice 2003, sborník příspěvků str. 187-192, 80-85960-59-1
9) Koziol J., *Optimalizace využití ložného prostoru*, Perners´ Contact 2003, Pardubice, sborník příspěvků str. 53.
10) Koziol J., *Bin Packing*, Infotrans 2004, Pardubice, sborník příspěvků str. 53.
11) Kučera L, *Kombinatoristické algoritmy*, SNTL Praha 1983
12) Levine J., Ducatelle F., *Ant Colony Optimisation and Local Search for Bin Packing and Cutting Stock Problems*, Centre for Intelligent Systems and their Applications, School of Informatics, University of Edinburgh 2004
13) Martello S., Toth P., *Knapsnack Problems: algorithms and computer implementations*, DEIS, University of Bologna, Biddles Ltd, Guilford 1990, ISBN 0 471 92420 2
14) Mathworld, on-line http://mathworld.wolfram.com/topics/PackingProblems.html
15) Novák, M., Faber, J., Kufudaki, O.: *Neuronové sítě a informační systémy živých organismů*, Praha, Grada 1993, 80-85424-95-9

16) Řezanková H., *Klasifikace pomocí shlukové analýzy*, VŠE Praha, 2003, on-line http://nb.vse.cz/~rezanka/Shlukova_analyza2003.pdf
17) Sweep S., *Three Dimensional Bin-Packing Issues and Solutions*, University of Minnesota, Morris, Computer Science Seminar, 2003
18) University of Arizona*, Bin Packing*, University of Arizona, 2003, on-line http://www.cs.arizona.edu/icon/oddsends/bpack/bpack.htm
19) *VDI 2702, Ladungssicherung auf Straßenfahrzeugen Zurrkräfte*, Düsseldorf 1990
20) Wikipedia, on-line www.wikipedia.com
21) Zelinka I., *Umělá inteligence I.*, Vitium, Brno 1998, ISBN: 80-214-1163-5
22) Žára J. a kolektiv, *Počítačová grafika – principy a algoritmy*, Edice Nestůjte za dveřmi, Grada, Praha 1992, ISBN: 80-85623-00-5

SEZNAM ZKRATEK

1D	jedna dimense („klasický“ Bin Packing problém)
2D	dvě dimense (plošná Bin Packing)
3D	tři dimense (prostorová úloha Bin Packing)
ACO	Ant Colony Optimisation
AWF	Almost Worst Fit algoritmus
BF	Best Fit algoritmus
BP	Bin Packing
cj	cenová jednotka
CP	Cutting Problem (řezný problém)
FF	First Fit algoritmus
KDD	Knowledge Discovery in Databases – dobývání znalostí z databází
KP	Knapsack Problem (úloha batohu)
Krit	kritérium účelové funkce
MTA	Martello-Tothův algoritmus
NF	Next Fit algoritmus
OP	omezující podmínka
PLM	překročená ložná míra
SW	Software
WF	Worst Fit algoritmus

SEZNAM GRAFŮ

SEZNAM OBRÁZKŮ

SEZNAM TABULEK