

Knihovna operačního výzkumu

Bc. Milan Hlinák

Diplomová práce
2015



Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2014/2015

ZADÁNÍ DIPLOMOVÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení: **Bc. Milan Hlinák**
Osobní číslo: **A13418**
Studijní program: **N3902 Inženýrská informatika**
Studijní obor: **Informační technologie**
Forma studia: **prezenční**

Téma práce: **Knihovna operačního výzkumu**
Téma anglicky: **An Operational Research Library**

Zásady pro vypracování:

1. Vypracujte literární rešerši na dané téma.
2. Popište matematicky nejznámější a nejstudovanější problematiky operačního výzkumu (Obchodní cestující, FlowShop, a další).
3. Ve zvoleném programovém prostředí (C/C++/Mathematica) vytvořte komplexní knihovnu operačního výzkumu.
4. Knihovnu otestujte na sadě benchmarkových dat.
5. Věnujte pozornost možnostem exportu výstupů knihovny/aplikace.
6. Zhodnoťte možnosti budoucího rozšíření a vylepšení vytvořené knihovny.

Rozsah diplomové práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. HILLIER, Frederick S. a Gerald J. LIEBERMAN. Introduction to operations research. 9th ed. New York: McGraw-Hill Higher Education, c2010, xxiv, 1047 s. ISBN 978-0-07-337629-5.
2. TAHA, Hamdy A. Operations research: an introduction. 9th ed. Upper Saddle River, N.J.: Prentice Hall, c2011, xxxi, 790 s. ISBN 978-0-13-255593-7.
3. FIALA, Petr. Operační výzkum: nové trendy. Praha: Professional Publishing, 2010, 239 s. ISBN 978-80-7431-036-2.
4. FÁBRY, Jan. Matematické modelování. Praha: Professional Publishing, 2011, 180 s. ISBN 978-80-7431-066-9.
5. ZIMOLA, Bedřich. Operační výzkum. 5. vyd. Zlín: Univerzita Tomáše Bati ve Zlíně, 2009, 168 s. ISBN 978-807-3188-788.
6. BRUCKER, Peter a Sigrid KNUST. Complex scheduling. 2nd ed. Heidelberg: Springer, c2012, x, 340 s. ISBN 978-3-642-23928-1.
7. RAVINDRAN, A. Ravi a Donald P. WARSING. Supply chain engineering: models and applications. Boca Raton: CRC Press, c2013, xxiv, 521 s. ISBN 978-1-4398-1198-6.

Vedoucí diplomové práce:

doc. Ing. Roman Šenkeřík, Ph.D.

Ústav informatiky a umělé inteligence

Datum zadání diplomové práce:

6. února 2015

Termín odevzdání diplomové práce:

15. května 2015

Ve Zlíně dne 6. února 2015



doc. Mgr. Milan Adámek, Ph.D.

děkan



L.S.



doc. Mgr. Roman Jašek, Ph.D.
ředitel ústavu

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně dne 15. května 2015

.....
podpis diplomanta

ABSTRAKT

Předmětem této diplomové práce je ve zvoleném programovém prostředí (C/C++/Mathematica) realizovat komplexní knihovnu nejznámějších a nejstudovanějších problematik operačního výzkumu. Tato knihovna bude zpracována ve formě účelových funkcí tak, aby mohly být tyto funkce dále v budoucnosti podrobeny testům pomocí různých algoritmů. Teoretická část je zaměřena na operační výzkum a matematický popis nejznámějších a nejstudovanějších problematik operačního výzkumu. Praktická část se zabývá návrhem a realizací knihovny operačního výzkumu, testováním na sadě benchmarkových dat a možnostmi budoucího rozšíření a vylepšení.

Klíčová slova: operační výzkum, matematické modelování, účelová funkce, softwarová knihovna

ABSTRACT

The subject of this master thesis is in the selected programming environment (C/C++/Mathematica) to implement a comprehensive library of the best known and the most studied issues of operational research. This library will be processed in the form of objective functions, so that these functions can be further subjected to tests in the future using various algorithms. The theoretical part is focused on operational research and mathematical description of the best known and the most studied issues of operational research. The practical part deals with the design and implementation of the operational research library, testing on a set of benchmark data and possibilities of future extensions and improvements.

Keywords: operational research, mathematical modeling, objective function, software library

Na tomto místě bych rád poděkoval vedoucímu diplomové práce panu doc. Ing. Romanu Šenkeříkovi, Ph.D. za odborné vedení, cenné rady a připomínky a ochotu při řešení dané problematiky. Dále bych chtěl poděkovat panu Ing. et Ing. Eriku Královi, Ph.D. za rady a připomínky k tvorbě softwarové knihovny. V neposlední řadě bych chtěl také poděkovat své rodině a přítelkyni za podporu při studiu a psaní této práce.

OBSAH

ÚVOD.....	10
I TEORETICKÁ ČÁST	11
1 OPERAČNÍ VÝZKUM	12
1.1 ÚVOD DO OPERAČNÍHO VÝZKUMU	12
1.2 POČÁTKY OPERAČNÍHO VÝZKUMU	13
1.3 DEFINICE OPERAČNÍHO VÝZKUMU	15
1.4 CHARAKTERISTIKA OPERAČNÍHO VÝZKUMU	15
1.5 FÁZE PŘI APLIKACI OPERAČNÍHO VÝZKUMU	17
1.6 DISCIPLÍNY OPERAČNÍHO VÝZKUMU	20
2 POPIS VYBRANÝCH PROBLÉMŮ OPERAČNÍHO VÝZKUMU	23
2.1 PROBLÉM BATOHU	23
2.1.1 Cenově nezávislý problém batohu	24
2.1.2 Ohraničený problém batohu	25
2.1.3 Problém batohu s vícenásobnou volbou.....	26
2.1.4 Vícerozměrný problém batohu.....	27
2.1.5 Vícenásobný problém batohu.....	28
2.2 PROBLÉM NAPLŇOVÁNÍ ZÁSOBNÍKU	29
2.3 LINEÁRNÍ PŘÍRAZOVACÍ PROBLÉM.....	31
2.3.1 Lineární součtový přiřazovací problém	31
2.3.2 Lineární úzkoprofilový přiřazovací problém	32
2.3.3 k -součtový přiřazovací problém	33
2.3.4 Vyvážený přiřazovací problém	33
2.4 KVADRATICKÝ PŘÍRAZOVACÍ PROBLÉM	34
2.4.1 Kvadratický úzkoprofilový přiřazovací problém	35
2.4.2 Kvadratický semi-přiřazovací problém.....	36
2.5 PROBLÉM ROZVRHOVÁNÍ PROUDOVÉ VÝROBY	37
2.5.1 Problém rozvrhování proudové výroby s neomezeným skladem	38
2.5.2 Problém rozvrhování proudové výroby s omezeným skladem	39
2.5.3 Problém rozvrhování proudové výroby s nulovým zpožděním	40
2.6 PROBLÉM OBCHODNÍHO CESTUJÍCÍHO.....	42
2.6.1 Symetrický problém obchodního cestujícího.....	42
2.6.2 Asymetrický problém obchodního cestujícího.....	43
2.6.3 Vícenásobný problém obchodního cestujícího	43
2.7 KAPACITNÍ ROZVOZNÍ PROBLÉM	43
2.7.1 Vzdálenostně omezený rozvozní problém	45
2.7.2 Rozvozní problém s časovými okny	45
2.7.3 Rozvozní problém se zpětným svozem.....	45
2.7.4 Rozvozní problém s vyzvednutím a doručením	46
II PRAKTICKÁ ČÁST	47
3 NÁVRH KNIHOVNY OPERAČNÍHO VÝZKUMU	48

3.1	POŽADAVKY NA KNIHOVNU OPERAČNÍHO VÝZKUMU	48
3.2	NÁVRH ZÁKLADNÍ STRUKTURY KNIHOVNY OPERAČNÍHO VÝZKUMU	49
3.2.1	Rozhraní <i>IData</i>	49
3.2.2	Třída obecného problému	50
3.2.3	Pomocné třídy	50
3.3	NÁVRH TŘÍD PROBLÉMŮ KNIHOVNY OPERAČNÍHO VÝZKUMU	51
3.3.1	Třída <i>KnapsackProblem</i>	51
3.3.2	Třída <i>BinPackingProblem</i>	53
3.3.3	Třída <i>LinearAssignmentProblem</i>	54
3.3.4	Třída <i>QuadraticAssignmentProblem</i>	55
3.3.5	Třída <i>FlowShopSchedulingProblem</i>	56
3.3.6	Třída <i>TravellingSalesmanProblem</i>	57
3.3.7	Třída <i>CapacitatedVehicleRoutingProblem</i>	58
3.4	NÁVRH STRUKTURY SOUBORŮ DAT PROBLÉMŮ	60
3.4.1	Soubor dat pro problém batohu	60
3.4.2	Soubor dat pro problém naplňování zásobníku	61
3.4.3	Soubor dat pro lineární přiřazovací problém	61
3.4.4	Soubor dat pro kvadratický přiřazovací problém	61
3.4.5	Soubor dat pro problém rozvrhování proudové výroby	62
3.4.6	Soubor dat pro problém obchodního cestujícího	62
3.4.7	Soubor dat pro kapacitní rozvozní problém	63
3.5	NÁVRH STRUKTURY SOUBORU DAT VÝSLEDKŮ	63
4	REALIZACE KNIHOVNY OPERAČNÍHO VÝZKUMU	65
4.1	REALIZACE TŘÍDY <i>FLOWSHOPSCHEDULINGPROBLEM</i>	65
4.1.1	Konstruktor <i>FlowShopSchedulingProblem</i>	65
4.1.2	Metoda <i>setData</i>	66
4.1.3	Metoda <i>generateData</i>	67
4.1.4	Metoda <i>readDataFromFile</i>	67
4.1.5	Metoda <i>writeDataToFile</i>	68
4.1.6	Metoda <i>showData</i>	68
4.1.7	Metoda <i>evaluate</i>	69
4.2	REALIZACE POMOCNÝCH TŘÍD	70
4.2.1	Třída <i>RandomHelper</i>	70
4.2.2	Třída <i>CoordinatesHelper</i>	70
4.2.3	Třída <i>ScheduleHelper</i>	71
5	TESTOVÁNÍ KNIHOVNY OPERAČNÍHO VÝZKUMU NA SADĚ BENCHMARKOVÝCH DAT	73
5.1	PROBLÉM BATOHU	73
5.2	PROBLÉM NAPLŇOVÁNÍ ZÁSOBNÍKU	74
5.3	LINEÁRNÍ PŘÍRAZOVACÍ PROBLÉM	74
5.4	KVADRATICKÝ PŘÍRAZOVACÍ PROBLÉM	75
5.5	PROBLÉM ROZVRHOVÁNÍ PROUDOVÉ VÝROBY	76
5.6	PROBLÉM OBCHODNÍHO CESTUJÍCÍHO	78
5.7	KAPACITNÍ ROZVOZNÍ PROBLÉM	79
6	DOKUMENTACE KNIHOVNY OPERAČNÍHO VÝZKUMU	80

6.1	PROGRAMOVÁ DOKUMENTACE.....	80
6.2	MANUÁL PRO POUŽITÍ KNIHOVNY	80
7	MOŽNOSTI BUDOUCÍHO ROZŠÍŘENÍ A VYLEPŠENÍ KNIHOVNY OPERAČNÍHO VÝZKUMU	81
8	APLIKACE S GRAFICKÝM UŽIVATELSKÝM ROZHRAŇÍM VYUŽÍVAJÍCÍ KNIHOVNY OPERAČNÍHO VÝZKUMU	82
	ZÁVĚR	83
	SEZNAM POUŽITÉ LITERATURY	85
	SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK	87
	SEZNAM OBRÁZKŮ	89
	SEZNAM TABULEK.....	90
	SEZNAM PŘÍLOH.....	91

ÚVOD

Každý aspekt lidského života je zásadně určen výsledkem rozhodnutí. Rozhodování představuje proces výběru jedné z možných alternativ, kterou lze dosáhnout požadovaného cíle. Zatímco soukromá rozhodnutí mohou být založena na emocích a osobním vkusu, profesio-
nální prostředí vyžaduje rozhodovací proces, který může být formalizován a ověřen nezá-
visle na zúčastněných jednotlivcích.

Operační výzkum je možné charakterizovat jako disciplínu, která aplikuje pokročilé analy-
tické metody k dosažení lepších rozhodnutí. Jeho počátek je obecně přisuzován vojenským
službám na počátku 2. světové války. Kvůli válečnému úsilí byla naléhavá potřeba přidělit
omezené prostředky na různé vojenské operace a činnosti v rámci jednotlivých operací
efektivním způsobem. Z tohoto důvodu byl povolán velký počet vědců, aby aplikovali vě-
decký přístup k řešení tohoto a dalších strategických a taktických problémů. Když válka
skončila, úspěch operačního výzkumu ve válečném úsilí vyvolal zájem o aplikaci operač-
ního výzkumu také v civilním sektoru.

Předmětem této práce je ve zvoleném programovém prostředí (C/C++/Mathematica) reali-
zovat komplexní knihovnu nejznámějších a nejstudovanějších problematik operačního vý-
zkumu. Knihovna bude zpracována ve formě účelových funkcí tak, aby mohly být tyto
funkce dále v budoucnosti podrobeny testům pomocí různých algoritmů.

V teoretické části práce je provedena literární rešerše na téma operační výzkum a matema-
tický popis vybraných problémů operačního výzkumu a jejich variant. Matematického po-
pisu problémů je dále využito pro implementaci účelových funkcí jednotlivých problémů
knihovny operačního výzkumu.

Praktická část práce se zabývá návrhem, realizací a testováním knihovny operačního vý-
zkumu. Návrh knihovny vychází ze stanovených požadavků kladených na knihovnu. Na
základě provedeného návrhu je následně provedena programová realizace knihovny ve
zvoleném programovém prostředí. Pro otestování jednotlivých problémů implementova-
ných v knihovně je využito standardních testovacích (benchmarkových) dat. Součástí prak-
tické části je také dokumentace a pojednání o možnostech budoucího rozšíření a vylepšení
realizované knihovny.

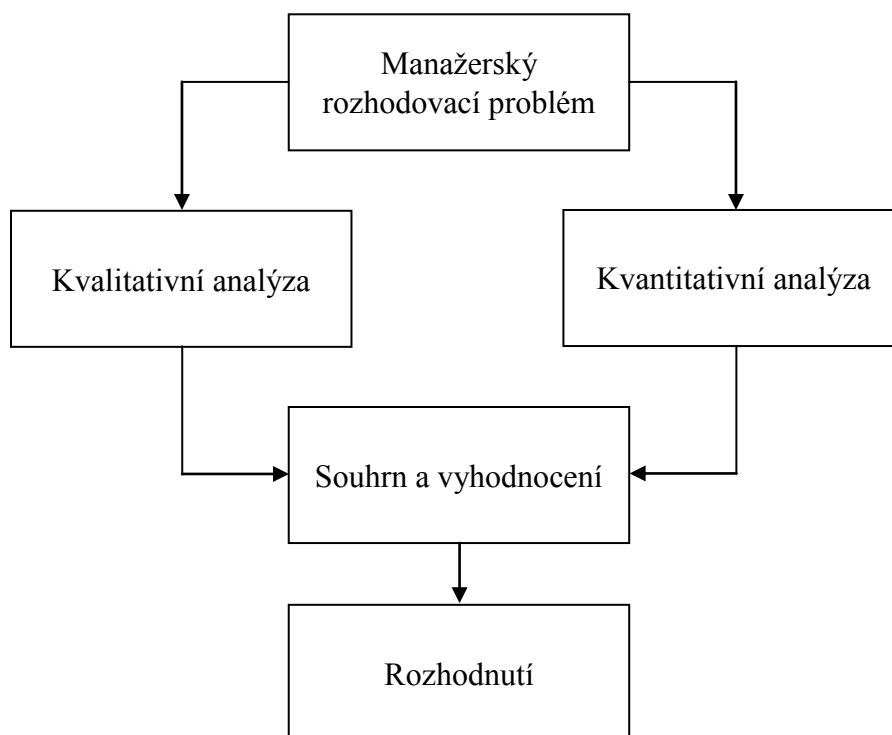
I. TEORETICKÁ ČÁST

1 OPERAČNÍ VÝZKUM

1.1 Úvod do operačního výzkumu

Rozhodování představuje proces výběru jedné z možných alternativ, kterou lze dosáhnout požadovaného cíle. Rozhodovací proces je činnost řídicích pracovníků, při níž usilují dojít k závěrům a rozhodnutím. Úspěch řídicích pracovníků při rozhodování závisí v podstatě na schopnosti vykonávat dvě funkce: analyzovat problém a učinit rozhodnutí. [1]

Manažerské rozhodování může vycházet z různých principů, které jsou závislé nejen na typu organizace, ale také na osobě samotného rozhodovatele. V současné době jsou kladeny vysoké nároky na rozhodování založené nejen na získaných zkušenostech, ale především na kvalifikovaných rozhodovacích analýzách, které jsou podpořeny konkrétními optimalizačními výstupy. Každý manažerský rozhodovací problém vyžaduje, aby byl zkoumán z hlediska kvantitativního a kvalitativního, jak je schematicky znázorněno na obrázku (Obr. 1). [2, s. 24]



Obr. 1. Rozdělení rozhodovacího procesu [2, s. 24]

Při kvantitativní analýze se využívají exaktní matematické metody, při kvalitativní analýze se využívají neformalizované metody. Potřebné informace musí být shromažďovány sou-

časně a posuzovány z hlediska daného problému. Cílem je přijmout rozhodnutí na základě těchto informačních zdrojů. [1]

Operační výzkum je dominantně zaměřen na hledání nejlepšího (optimálního) řešení vycházejícího převážně z kvantifikovaných údajů, což odpovídá části kvantitativní analýza. Tato optimalizace ale nemůže být prováděna odděleně a musí vždy respektovat určité podmínky kvalitativní části řešeného problému. [2, s. 24]

Metody operačního výzkumu je tedy možné zařadit mezi nástroje managementu určené k hledání optimálního řešení reálného manažerského problému. Tyto metody přitom využívají celé spektrum klasických vědních disciplín jako je např. matematika, statistika, ale rovněž novější vědecké obory jako je např. teorie rozhodování. Předmětem zkoumání metod operačního výzkumu jsou nejen vlastní optimalizované systémy, ale také jednotlivá rozhodnutí manažerů řídících provoz systémů. [2, s. 24]

Operační výzkum je relativně mladý vědní obor. V české literatuře se můžeme setkat s názvy jako jsou např. operační výzkum, operační analýza apod. V zahraniční literatuře se můžeme setkat s názvy jako jsou např. operations research, operational research, management science, decision science apod.

1.2 Počátky operačního výzkumu

Od příchodu průmyslové revoluce zažil svět pozoruhodný nárůst velikosti a složitosti organizací. Nedílnou součástí této revoluční změny byl obrovský nárůst v dělbě práce a členění odpovědnosti za řízení v těchto organizacích. Výsledky byly velkolepé, avšak tento nárůst specializace vytvořil nové problémy, které se stále vyskytují v mnoha organizacích. Jedním z problémů je tendence pro mnoho složek organizace růst do relativně autonomních částí s vlastními cíli a systémy hodnot, čímž ztrácejí ze zřetele, jak se jejich činnosti a cíle shodují s činnostmi a cíli celkové organizace. Co je nejlepší pro jednu složku často poškozuje jinou, takže složky mohou skončit se zkříženými cíli. Souvisejícím problémem je, že s růstem složitosti a specializace organizace se stává stále obtížnější přidělovat dostupné prostředky na různé činnosti takovým způsobem, který je nejefektivnější pro organizaci jako celek. Tyto druhy problémů a potřeba najít lepší způsob jejich řešení zajistily prostředí pro vznik operačního výzkumu. [12, s. 1]

Kořeny operačního výzkumu lze vysledovat mnoho desetiletí zpět, kdy byly provedeny první pokusy použití vědeckého přístupu při managementu organizací. Nicméně začátek

činnosti s názvem operační výzkum je obecně přisuzován vojenským službám na počátku 2. světové války. Kvůli válečnému úsilí byla naléhavá potřeba přidělit omezené prostředky na různé vojenské operace a činnosti v rámci jednotlivých operací efektivním způsobem. Z tohoto důvodu byl ve Velké Británii a USA povolán velký počet vědců, aby aplikovali vědecký přístup k řešení tohoto a dalších strategických a taktických problémů. Ve skutečnosti byli požádáni, aby provedli výzkum (vojenských) operací. Tyto týmy vědců byly prvními týmy operačního výzkumu. [12, s. 1]

Když válka skončila, úspěch operačního výzkumu ve válečném úsilí vyvolal zájem o aplikaci operačního výzkumu také v civilním sektoru. Jak probíhal poválečný průmyslový rozvoj, problémy způsobené rostoucí složitostí a specializací organizací se opět dostávaly do popředí. Stávalo se zřejmým, že jde v podstatě o stejné problémy jako ve válce, ale v jiném kontextu. Během 50. let 20. století bylo představeno využití operačního výzkumu v různých organizacích v oblasti obchodu, průmyslu a vlády. Brzy následovalo rychlé rozšíření operačního výzkumu. [12, s. 2]

Jedním z faktorů, který hrál důležitou roli v rapidním růstu operačního výzkumu v tomto období, byl značný pokrok, který byl proveden na počátku zlepšování technik operačního výzkumu. Po válce bylo mnoho vědců motivováno k výzkumu týkající se oblasti operačního výzkumu, což mělo za následek významné pokroky. Příkladem je simplexová metoda pro řešení úloh lineárního programování vyvinutá Georgem Dantzigem v roce 1947. Mnoho standardních nástrojů operačního výzkumu, jako je lineární programování, dynamické programování, teorie zásob a další, byly poměrně dobře vyvinuty před koncem 50. let 20. století. [12, s. 2]

Dalším důležitým faktorem, který ovlivnil rozvoj operačního výzkumu, byl rozvoj výpočetní techniky. Složité problémy, které jsou řešeny v rámci operačního výzkumu, obvykle vyžadují velké množství výpočtů pro co nejefektivnější řešení. Provádění těchto výpočtů ručně tedy nepřípadá v úvahu. Proto vývoj počítačů, s jejich schopností vykonávat výpočty mnohonásobně rychleji než člověk, měl obrovský přínos pro operační výzkum. Další pokrok přišel s vývojem stále více výkonných osobních počítačů v doprovodu s dobrými softwarovými balíčky pro operační výzkum. To přineslo využívání operačního výzkumu mnohem větším počtem lidí a tento pokrok se dále zrychlil v 90. letech 20. století a do 21. století. V současné době mají lidé snadný přístup k softwaru pro operační výzkum. V důsledku toho je celá řada počítačů běžně používána k řešení problémů operačního výzkumu, včetně některých problémů velkých rozměrů. [12, s. 2]

1.3 Definice operačního výzkumu

Stejně jako existuje větší množství používaných názvů pro vědní obor operační výzkum, stejně tak existuje i celá řada různých definic tohoto oboru. Každá literatura uvádí odlišnou definici, používá jiná slova či slovní spojení. Všechny definice ale narážejí na jednu podstatnou skutečnost, kterou je rozmanitost problémů a odpovídajících disciplín. Z tohoto důvodu přesnou definici operačního výzkumu jednoduše podat nelze.

Jak uvádí Murthy [4], každý jednotlivec může definovat operační výzkum svým způsobem v závislosti na jeho cíli. Každá definice může vysvětlovat některou charakteristiku operačního výzkumu, ale žádná z nich nevysvětlí nebo nepodá úplný obraz o operačním výzkumu.

Jako vybrané definice operačního výzkumu lze uvést např.:

„Operační výzkum je aplikace vědeckých metod, technik a nástrojů k nalezení optimálního řešení problému systému.“ [4, s. 8]

„Operační výzkum je možné charakterizovat jako vědní disciplínu nebo spíše soubor relativně samostatných disciplín, které jsou zaměřeny na analýzu různých typů rozhodovacích problémů.“ [5, s. 9]

„Operační výzkum je disciplínou, která aplikuje pokročilé analytické metody k dosažení lepších rozhodnutí.“ [6]

1.4 Charakteristika operačního výzkumu

Jako již bylo uvedeno, metody operačního výzkumu je možné charakterizovat jako manažerské nástroje pro nalezení nejlepšího neboli optimálního řešení zkoumaného problému. S uplatněním metod operačního výzkumu je možné se setkat ve všech oblastech, kde se jedná o analýzu a koordinaci provádění operací v rámci nějakého systému. Jak uvádí např. Zimola [7], pro úspěšnou aplikaci metod operačního výzkumu do reálného procesu jsou charakteristické tři základní principy - systémový přístup, interdisciplinární týmový přístup a modelová technika.

Systémový přístup

Při řešení úloh operačního výzkumu se vychází z předpokladu, že chování libovolného prvku systému, který je předmětem zájmu, nelze posuzovat izolovaně, jelikož každý prvek určitým způsobem ovlivňuje ostatní prvky. Přitom je nutné brát v úvahu, že ne každá z

těchto vazeb je podstatná a ne všechny vlivy je možné zjistit. Základem systémového přístupu při řešení problému je vyhledávání vzájemných vazeb při zkoumání kterékoliv části systému, přitom se přihlíží k působení okolí. Zpravidla se vychází z jednoduché formulace úlohy, která se případně rozšíří o další vazby. [7, s. 7, 8]

Interdisciplinární týmový přístup

Praktické zkušenosti potvrzují, že řešení složitých rozhodovacích úloh je možné získat pouze s použitím poznatků řady vědních disciplín. Proto se v operačním výzkumu zapojují do řešení problémů specialisté různých oborů a zaměření. Interdisciplinární přístup pak umožňuje zkoumání úlohy z různých hledisek, jelikož každý specialista uplatňuje při řešení problému svůj vlastní přístup. [7, s. 8]

Modelová technika

Základním nástrojem operačního výzkumu je matematické modelování. Pokud je nějaký systém analyzován pomocí operačního výzkumu, potom tato analýza využívá model tohoto systému. Pomocí modelu zobrazujeme podstatné vlastnosti modelovaného systému, tj. systému, který je předmětem zájmu. Modelem můžeme rozumět zjednodušení reality. Není samozřejmě možné popsat realitu do všech podrobností. Z tohoto důvodu je nutné se při analýze reálného systému soustředit na ty části, které jsou z hlediska cíle analýzy důležité. [8, s. 9]

Univerzální postup pro vytvoření modelu neexistuje. Při vytváření modelu je však důležité brát v úvahu, že pokud model zachycuje příliš mnoho vlastností modelovaného systému, pak tento model bude kvalitní, ale bude těžké jej sestavit a pracovat s ním. Naopak pokud model zachycuje příliš málo vlastností, je zpravidla méně přesný, ale může být snáze zvládnutelný. Při vytváření modelů je tedy nutné najít určitý kompromis mezi úrovní zachycení vlastností modelovaného systému a řešitelností úlohy vyjádřené daným modelem. [9, s. 5]

Typickým znakem modelů operačního výzkumu je především skutečnost, že mají charakter optimalizačních modelů. V optimalizačních úlohách hledáme řešení, které je nejlepší mezi všemi možnými přípustnými řešeními. Obvykle je dána množina řešení, tzv. prostor řešení, a seznam podmínek, tzv. omezující podmínky. Množina přípustných řešení je pak tvořena řešeními, která splňují všechny omezující podmínky. Které řešení je lepší a které horší je určováno pomocí účelové funkce. Účelová funkce je matematickým modelem řešeného problému a její hodnota udává kvalitu řešení. Optimální řešení je pak takové pří-

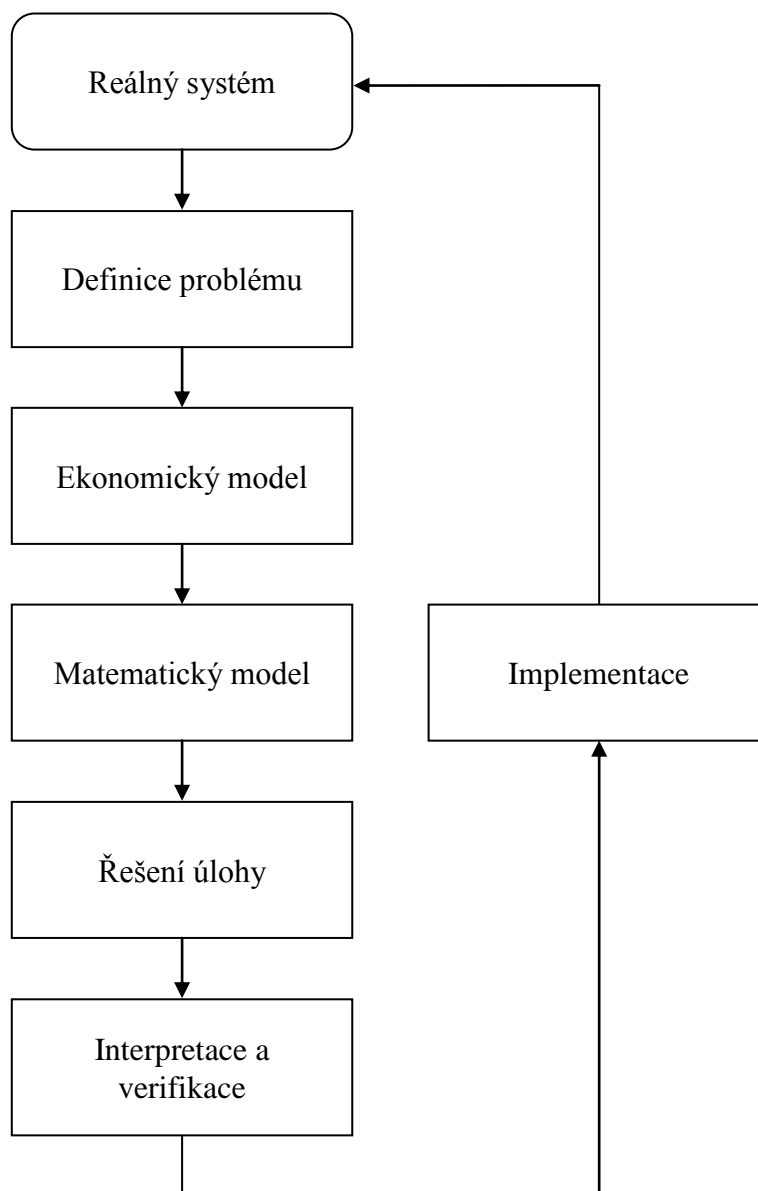
pustné řešení, které má mezi všemi přípustnými řešeními nejlepší hodnotu účelové funkce. Zde záleží na charakteru úlohy - v případě minimalizační úlohy budeme hledat řešení s nejmenší hodnotou účelové funkce, naopak v případě maximalizační úlohy budeme hledat řešení s největší hodnotou účelové funkce. Úloha může mít také několik optimálních řešení se stejnou hodnotou účelové funkce. [9, s. 7]

1.5 Fáze při aplikaci operačního výzkumu

Studium operačního výzkumu má své kořeny v týmové práci, kde analytici a klienti pracují bok po boku. Odbornost analytiků operačního výzkumu v modelování musí být doplněna o zkušenosti a spolupráci klienta, pro kterého se studie provádí. [10, s. 8]

Jako rozhodovací nástroj je operační výzkum jak věda, tak umění. Je to věda na základě matematických postupů, které ztělesňuje, a je to umění, protože úspěch fází vedoucích k řešení matematického modelu závisí do značné míry na kreativitě a zkušenosti týmu operačního výzkumu. [10, s. 8]

Při aplikaci některého odvětví operačního výzkumu pro řešení rozhodovacího problému je možné rozlišit několik základních fází, jak je uvedeno na obrázku (Obr. 2).



Obr. 2. Fáze při aplikaci operačního výzkumu [5, s. 11]

Definice problému

Prvním podstatným krokem aplikace modelů operačního výzkumu je rozpoznání problému v rámci reálného systému. Po rozpoznání problému je nutné dokázat tento problém jasně a přesně definovat pro potřeby matematického modelování. [5, s. 10]

Ekonomický model

Ekonomický model je možné charakterizovat jako zjednodušený popis reálného systému, který obsahuje pouze nejpodstatnější prvky a vazby mezi nimi, s ohledem na analyzovaný problém. Ekonomický model by měl obsahovat především:

- cíl analýzy,
- popis procesů, které v systému probíhají,
- popis činitelů ovlivňujících provádění procesů,
- popis vztahu mezi procesy, činiteli a cílem analýzy,

dle [5, s. 10, 11].

Cílem analýzy se rozumí určení cílového stavu modelovaného systému. Cíl analýzy souvisí s hlavním impulzem k provedení analýzy systému a většinou vychází z nespokojenosti se současným stavem. Jako proces můžeme chápat reálnou aktivitu, která probíhá v systému s určitou intenzitou, která má vliv na cíl analýzy. Procesy nemohou být v reálných úlohách prováděny s neomezenou intenzitou, jelikož jejich realizace je ovlivněna řadou různých činitelů.

Pokud uvažujeme, že procesem je výroba určitého výrobku, činitelem spotřeba určitého materiálu a cílem maximalizace zisku, pak je třeba určit, jaké množství materiálu se spotřebuje při výrobě výrobku a jaký bude odpovídající zisk. [5, s. 11, 12]

Matematický model

Formulace matematického modelu vychází z ekonomického modelu. Aby bylo možné daný problém řešit, je třeba převést ekonomický model na model matematický. Matematický model obsahuje stejné prvky jako model ekonomický, ale v jiném vyjádření - cíl analýzy je vyjádřen jako účelová funkce, procesy jako rozhodovací proměnné, činitelé jako omezující podmínky, vztahy mezi procesy, činiteli a cílem analýzy jako neřiditelné proměnné. [5, s. 12]

Řešení úlohy

Řešení úlohy je obecně nejjednodušší fází aplikace operačního výzkumu. Na rozdíl od ostatních fází, které jsou spíše uměním, je řešení úlohy spíše technickou záležitostí. Pro řešení lze využít různé metody a postupy navržené v různých odvětvích operačního výzkumu. [5, s. 12]

Interpretace a verifikace

Interpretace a verifikace výsledků je velmi důležitou fází aplikace operačního výzkumu. Interpretací výsledků se rozumí slovní vyjádření či vysvětlení výsledků získaných řešením úlohy. Samotná interpretace však nestačí. Je nutné verifikovat výsledky a tím ověřit, že

sestavení ekonomického modelu a následně matematického modelu bylo správné. Pokud při sestavování modelu došlo k opomenutí některých důležitých skutečností, potom řešení modelu může být sice optimální v rámci tohoto modelu, nicméně v praxi může být nepoužitelné. [5, s. 13]

Implementace

Po úspěšné fázi interpretace a verifikace výsledků lze přistoupit k implementaci v rámci reálného systému. Cílem implementace je zlepšení fungování systému s ohledem na sledovaný cíl. Po implementaci výsledků by měla následovat ještě kontrola, zda provedená změna byla pro reálný systém opravdu přínosná. [5, s. 13]

1.6 Disciplíny operačního výzkumu

Modely operačního výzkumu jsou velmi různorodé a zabývají se různými oblastmi. Z tohoto důvodu je potřeba specifických přístupů k řešení jednotlivých tříd problémů. V operačním výzkumu není jediná obecná technika, jak vyřešit všechny matematické modely, které mohou nastat v praxi. Místo toho, druh a složitost matematického modelu určují povahu metod řešení. [10, s. 4]

Dále jsou uvedeny nejznámější a nejpoužívanější disciplíny operačního výzkumu, jak uvádí např. Jablonský [5].

1. Matematické programování

Matematické programování se zabývá řešením obecné úlohy nalezení maximální resp. minimální hodnoty účelové funkce n proměnných při splnění soustavy omezujících podmínek. Podle typu účelové funkce a jednotlivých omezení lze úlohy matematického programování rozdělit do dvou základních skupin - lineární a nelineární programování. Matematický model úlohy matematického programování lze zapsat ve tvaru maximalizace resp. minimalizace účelové funkce

$$z = f(x_1, x_2, \dots, x_n)$$

za podmínek

$$g_i(x_1, x_2, \dots, x_n) \geq 0 \quad i = 1, 2, \dots, m$$

$$x_j \geq 0 \quad j = 1, 2, \dots, n$$

dle [7, s. 16].

Je-li kritériální funkce lineární a všechny rovnice a nerovnice jsou také lineární, pak jde o úlohu lineárního programování, v opačném případě jde o úlohu nelineárního programování.

2. Teorie grafů

Teorie grafů se zabývá analýzou rozhodovacích problémů, jejichž model je vyjádřen ve formě tzv. grafu. Jako grafy jsou chápány objekty, které jsou tvořeny množinou uzlů a množinou hran. Uzly většinou znázorňují prvky v reálném systému a hrany vztahy mezi nimi. Pomocí těchto grafů je možné znázorňovat různé reálné systémy. [8, s. 61]

3. Teorie zásob

Teorie zásob se zabývá způsoby řízení zásobovacího procesu a optimalizací objemu zásob s ohledem především na minimalizaci nákladů s tím souvisejících. Cílem modelů řízení zásob je dát odpověď na otázku kdy a kolik výrobků na sklad objednávat nebo vyrábět. Přitom je třeba najít ekonomicky výhodný poměr mezi náklady na skladování a ztrátami způsobnými nedostatkem zásob. [5, s. 15]

4. Teorie hromadné obsluhy

Teorie hromadné obsluhy se zabývá zkoumáním systémů, ve kterých se vyskytují dva základní typy jednotek - požadavky a obslužné linky. Požadavky vstupují do systému a vyžadují obsluhu, kterou realizují obslužné linky. S realizací požadavků souvisí také tvorba front, z čehož plyne alternativní název - teorie front. [5, s. 15]

5. Teorie her

Teorie her je disciplínou, která analyzuje široké spektrum konfliktních rozhodovacích situací. Tyto situace mohou nastat kdekoliv, kde dochází ke střetu zájmů. Teorie her se snaží tyto konfliktní situace nejen analyzovat, ale také se snaží nalézt co nejlepší strategie pro účastníky rozhodovacích problémů, kteří si vzájemně konkurují. Vychází se z toho, že řadu rozhodovacích situací s více než jedním účastníkem je možné popsat jako hru, kde jednotliví účastníci mají určité strategie, na nichž závisí jejich výhra. [5, s. 16]

6. Teorie obnovy

Teorie obnovy se zabývá zkoumáním systémů, ve kterých po určité době provozu dochází k selhání jednotek. Tyto jednotky musí být následně opraveny resp. nahrazeny. Cílem je odhadnout věkovou strukturu a počet selhaných jednotek v čase, aby se preventivní výměnou v teoreticky odhadnuté lhůtě předešlo případným větším ztrátám. [5, s. 16]

7. Vícekriteriální rozhodování

Vícekriteriální rozhodování se zabývá analýzou rozhodovacích úloh, ve kterých jsou varianty pro rozhodování posuzovány podle několika hodnotících kritérií současně, přičemž varianta hodnocená podle jednoho kritéria zpravidla nebývá nejlépe hodnocená podle kritéria jiného. Cílem rozhodování je vybrat variantu, která je podle daných kritérií ohodnocena nejlépe. [5, s. 14]

8. Simulace

Simulace lze definovat jako technika, která napodobuje chování reálného systému a jeho vývoj v čase. Simulační model se obvykle sestavuje na základě předpokladů o procesech, které probíhají v systému, a vyjadřuje vztahy mezi prvky systému. [8, s. 139]

Kromě výše uvedených disciplín operačního výzkumu je však nutné zmínit také přístup k řešení problémů, který se nazývá soft computing (SC). Mezi hlavní zástupce SC zahrnujeme fuzzy logiku, neuronové sítě a genetické algoritmy. SC je fúze metodik, které byly navrženy pro modelování a umožnění řešení složitých reálných problémů, které je velmi složité přesně modelovat nebo jejich exaktní řešení by si vyžádalo extrémní požadavky na výpočetní prostředky a čas. Hlavním cílem SC je využít toleranci pro nepřesnost a neurčitost k dosažení flexibility, robustnosti a nízkonákladového řešení. U SC není důležitý model mechanismů, kterými se řídí chování systému (tj. „vnitřní znalost“), důležitá je existence znalostí o chování systému (tj. „vnější znalost“). [11, s. 5-9]

2 POPIS VYBRANÝCH PROBLÉMŮ OPERAČNÍHO VÝZKUMU

Tato kapitola se zabývá popisem nejznámějších a nejstudovanějších problematik operačního výzkumu. U každého problému je uvedena stručná charakteristika a matematický popis problému. Kromě českého názvu problému je uveden také anglický ekvivalent, se kterým je možné se setkat v zahraniční literatuře zabývající se problematikou operačního výzkumu.

2.1 Problém batohu

Problém batohu (Knapsack problem, KP) patří mezi nejznámější problémy kombinatorické optimalizace. KP řeší otázku výběru podmnožiny předmětů specifikovaných cenou a hmotností, které mají být umístěny do batohu s omezenou kapacitou tak, aby celková cena předmětů umístěných do batohu byla maximalizována a celková hmotnost předmětů umístěných do batohu nepřekročila kapacitu batohu.

Jako příklad může být uvažován horolezec, který si balí batoh na horskou túru. Horolezec má k dispozici velké množství předmětů, které by mohly být užitečné na jeho túře, a musí se rozhodnout, které předměty si vezme s sebou. Každý předmět má určitou hmotnost a určitou míru užitku. Samozřejmě hmotnost každého předmětu, který je umístěn do batohu, zvyšuje celkové zatížení, které horolezec musí nést. Z pochopitelných důvodů chce horolezec omezit celkovou hmotnost svého batohu, a proto stanoví maximální možné zatížení hodnotou kapacity batohu. [12, s. 2]

KP může být modelován jako maximalizace účelové funkce

$$z = \sum_{j=1}^n p_j x_j \quad (1)$$

za podmínky

$$\sum_{j=1}^n w_j x_j \leq c \quad (2)$$

kde

- n je počet předmětů,
- p_j je cena předmětu j ,
- w_j je hmotnost předmětu j ,

- c je kapacita batohu,
- x_j je binární proměnná, jejíž hodnota je 1, jestliže předmět j je vybrán, jinak je její hodnota 0,

dle [12, s. 2, 3].

Podmínka (2) zajišťuje, že hmotnost vybraných předmětů nepřekročí kapacitu batohu.

Dále se předpokládá, že

- hmotnost w_j a cena p_j každého předmětu j a kapacita batohu c jsou kladné hodnoty

$$w_j, p_j, c > 0 \quad j = 1, 2, \dots, n \quad (3)$$

- hmotnost w_j každého předmětu j je menší nebo rovna kapacitě batohu c

$$w_j \leq c \quad j = 1, 2, \dots, n \quad (4)$$

Tyto předpoklady jsou platné také pro následující varianty problému. [12, s. 10]

2.1.1 Cenově nezávislý problém batohu

Cenově nezávislý problém batohu (Subset sum problem, SSP) je zvláštním případem KP, kdy cena a hmotnost jsou ekvivalentní pro každý předmět. Úkolem je výběr takové podmnožiny množiny předmětů, aby celková hmotnost byla maximalizována, ale nepřekročila kapacitu batohu. [13, s. 105]

SSP může být modelován jako maximalizace účelové funkce

$$z = \sum_{j=1}^n w_j x_j \quad (5)$$

za podmínky

$$\sum_{j=1}^n w_j x_j \leq c \quad (6)$$

kde

- n je počet předmětů,
- w_j je hmotnost předmětu j ,
- c je kapacita batohu,

- x_j je binární proměnná, jejíž hodnota je 1, jestliže předmět j je vybrán, jinak je její hodnota 0,

dle [13, s. 105].

Podmínka (6) zajišťuje, že hmotnost vybraných předmětů nepřekročí kapacitu batohu.

2.1.2 Ohraničený problém batohu

Ohraničený problém batohu (Bounded knapsack problem, BKP) je zobecněním KP, kde se předpokládá omezený počet identických kopií každého předmětu. Úkolem je výběr předmětů každého typu tak, aby celková cena byla maximalizována a celková hmotnost nepřekročila kapacitu batohu. [12, s. 185]

BKP může být modelován jako maximalizace účelové funkce

$$z = \sum_{j=1}^n p_j x_j \quad (7)$$

za podmínek

$$\sum_{j=1}^n w_j x_j \leq c \quad (8)$$

$$x_j \in \mathbb{Z}_0^+ \quad x_j \leq b_j \quad j = 1, 2, \dots, n \quad (9)$$

kde

- n je počet typů předmětů,
- p_j je cena předmětu j ,
- w_j je hmotnost předmětu j ,
- c je kapacita batohu,
- b_j je počet identických kopií předmětu j ,
- x_j je celočíselná proměnná udávající počet vybraných předmětů typu j ,

dle [12, s. 185].

Podmínka (8) zajišťuje, že hmotnost vybraných předmětů nepřekročí kapacitu batohu.

Podmínka (9) zajišťuje, že počet vybraných předmětů typu j může být v rozmezí od 0 do b_j .

Předpoklad (4) je upraven do tvaru

$$b_j \leq \left\lfloor \frac{c}{w_j} \right\rfloor \quad j=1,2,\dots,n \quad (10)$$

dle [12, s. 186].

Zvláštním případem BKP je neohraničený problém batohu (unbounded knapsack problem, UKP), kde se předpokládá neomezený počet identických kopií každého předmětu. Podmínka (9) je tedy upravena do tvaru

$$x_j \in Z_0^+ \quad j=1,2,\dots,n \quad (11)$$

dle [12, s. 211].

2.1.3 Problém batohu s vícenásobnou volbou

Problém batohu s vícenásobnou volbou (Multiple-choice knapsack problem, MCKP) je zobecněním KP, kde množina předmětů je rozdělena do tříd. Binární volba výběru předmětu je nahrazena výběrem právě jednoho předmětu z každé třídy předmětů. Úkolem je vybrat právě jednoho předmětu z každé třídy předmětů tak, aby celková cena byla maximalizována a celková hmotnost nepřekročila kapacitu batohu. [12, s. 317]

MCKP může být modelován jako maximalizace účelové funkce

$$z = \sum_{i=1}^m \sum_{j \in N_i} p_{ij} x_{ij} \quad (12)$$

za podmínek

$$\sum_{i=1}^m \sum_{j \in N_i} w_{ij} x_{ij} \leq c \quad (13)$$

$$\sum_{j \in N_i} x_{ij} = 1 \quad i=1,2,\dots,m \quad (14)$$

kde

- m je počet tříd předmětů,
- N_i je i -tá třída předmětů,
- p_{ij} je cena předmětu j třídy předmětů i ,
- w_{ij} je hmotnost předmětu j třídy předmětů i ,

- c je kapacita batohu,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže předmět j třídy předmětů i je vybrán, jinak je její hodnota 0,

dle [12, s. 317].

Podmínka (13) zajišťuje, že hmotnost vybraných předmětů nepřekročí kapacitu batohu.

Podmínka (14) zajišťuje, že z každé třídy předmětů bude vybrán právě jeden předmět.

Předpoklady (4) a (5) jsou upraveny do tvaru

$$w_{ij} > 0, p_{ij} > 0, c > 0 \quad i = 1, 2, \dots, m \quad j \in N_i \quad (15)$$

$$w_{ij} \leq c \quad i = 1, 2, \dots, m \quad j \in N_i \quad (16)$$

dle [12, s. 317].

2.1.4 Vícerozměrný problém batohu

Vícerozměrný (d -rozměrný) problém batohu (d -dimensional knapsack problem, d -KP) je zobecněním KP, kde se k omezení hmotnosti přidávají další omezení. Každý předmět je specifikován cenou a d parametry. Batoh má d omezení. Úkolem je výběr takové podmnožiny množiny předmětů, aby celková cena byla maximalizována a žádný součet i -tých parametrů předmětů nepřekročil odpovídající i -té omezení batohu. [12, s. 235]

d -KP může být modelován jako maximalizace účelové funkce

$$z = \sum_{j=1}^n p_j x_j \quad (17)$$

za podmínek

$$\sum_{j=1}^n w_{ij} x_j \leq c_i \quad i = 1, 2, \dots, d \quad (18)$$

kde

- n je počet předmětů,
- d je počet rozměrů (parametrů, omezení),
- p_j je cena předmětu j ,
- w_{ij} je i -tý parametr předmětu j ,

- c_i je i -té omezení batohu,
- x_j je binární proměnná, jejíž hodnota je 1, jestliže předmět j je vybrán, jinak je její hodnota 0,

dle [12, s. 235].

Soustava podmínek (18) zajišťuje, že žádný součet i -tých parametrů vybraných předmětů nepřekročí odpovídající i -té omezení batohu.

Předpoklady (4) a (5) jsou upraveny do tvaru

$$w_{ij} > 0, p_j > 0, c_i > 0 \quad j = 1, 2, \dots, n \quad i = 1, 2, \dots, d \quad (19)$$

$$w_{ij} \leq c_i \quad j = 1, 2, \dots, n \quad i = 1, 2, \dots, d \quad (20)$$

dle [12, s. 235, 236].

Pro parametry předmětů je dovoleno $w_{ij} = 0$ pro určité i, j , pokud platí $\sum_{i=1}^d w_{ij} > 0$ pro všechny předměty j , tj. alespoň jeden parametr předmětu nabývá hodnoty větší než 0. [12, s. 235, 236]

2.1.5 Vícenásobný problém batohu

Vícenásobný problém batohu (Multiple knapsack problem, MKP) je zobecněním KP, kde se uvažuje m batohů, které mohou mít odlišné kapacity. Úkolem je vybrat m disjunktních podmnožin předmětů tak, aby celková cena byla maximalizována a každá podmnožina mohla být přiřazena do jednoho batohu, aniž by celková hmotnost předmětů této podmnožiny překročila kapacitu batohu.

MKP můžeme formulovat následovně: Je dána množina n předmětů. Každý předmět j je specifikován cenou p_j a hmotností w_j . Dále je dána množina m batohů. Každý batoh i je specifikován kapacitou c_i .

MKP může být modelován jako maximalizace účelové funkce

$$z = \sum_{i=1}^m \sum_{j=1}^n p_j x_{ij} \quad (21)$$

za podmínek

$$\sum_{j=1}^n w_j x_{ij} \leq c_i \quad i = 1, 2, \dots, m \quad (22)$$

$$\sum_{i=1}^m x_{ij} \leq 1 \quad j = 1, 2, \dots, n \quad (23)$$

kde

- n je počet předmětů,
- m je počet batohů,
- p_j je cena předmětu j ,
- w_j je hmotnost předmětu j ,
- c_i je kapacita batohu i ,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže předmět j je umístěn do batohu i , jinak je její hodnota 0,

dle [12, s. 285].

Podmínka (22) zajišťuje, že hmotnost předmětů umístěných do batohu nepřekročí kapacitu tohoto batohu. Podmínka (23) zajišťuje, že každý předmět se vyskytne nanejvýš jednou ve všech batozích.

Předpoklady (4) a (5) jsou upraveny do tvaru

$$w_j > 0, p_j > 0, c_i > 0 \quad j = 1, 2, \dots, n \quad i = 1, 2, \dots, m \quad (24)$$

$$w_{\min} \leq c_{\min} \quad w_{\max} \leq c_{\max} \quad (25)$$

kde

- w_{\min} (w_{\max}) je minimální (maximální) hmotnost předmětu,
- c_{\min} (c_{\max}) je minimální (maximální) kapacita batohu,

dle [12, s. 286].

2.2 Problém naplňování zásobníku

Problém naplňování zásobníku (Bin packing problem, BPP) řeší otázku umístění předmětů specifikovaných hmotností do zásobníků s omezenou kapacitou tak, aby všechny předměty byly umístěny do zásobníků a celkový počet potřebných zásobníků byl minimalizován.

BPP může být interpretován jako vícenásobný cenově nezávislý problém batohu, kde všechny batohy resp. zásobníky mají stejnou kapacitu a všechny předměty musí být umístěny do zásobníků. [13, s. 221]

BPP může být modelován jako minimalizace účelové funkce

$$z = \sum_{i=1}^n y_i \quad (26)$$

za podmínky

$$\sum_{j=1}^n w_j x_{ij} \leq cy_i \quad i = 1, 2, \dots, n \quad (27)$$

kde

- n je počet předmětů,
- w_j je hmotnost předmětu j ,
- c je kapacita zásobníku,
- y_i je binární proměnná, jejíž hodnota je 1, jestliže zásobník i je použit, jinak je její hodnota 0,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže předmět j je umístěn do zásobníku i , jinak je její hodnota 0,

dle [13, s. 221].

Podmínka (27) zajišťuje, že hmotnost předmětů umístěných do zásobníku nepřekročí kapacitu zásobníku.

Dále se předpokládá, že

- hmotnost w_j každého předmětu j a kapacita zásobníku c jsou kladné hodnoty

$$w_j, c > 0 \quad j = 1, 2, \dots, n \quad (28)$$

- hmotnost w_j každého předmětu j je menší nebo rovna kapacitě zásobníku c

$$w_j \leq c \quad j = 1, 2, \dots, n \quad (29)$$

dle [13, s. 221, 222].

2.3 Lineární přiřazovací problém

Problémy přiřazení se zabývají otázkou, jak přiřadit n objektů (např. úloh) k n jiným objektům (např. stroje) nejlepším možným způsobem. Existují různé způsoby popisu přiřazení. Přiřazení můžeme popsat jako bijektivní zobrazení mezi dvěma konečnými množinami o n prvcích. Identifikováním množin získáme reprezentaci přiřazení pomocí permutace. [14, s. 1]

Lineární přiřazovací problém (Linear assignment problem, LAP) je přiřazovací problém s lineární účelovou funkcí. Příkladem je situace, kdy chceme přiřadit n úloh k n strojům nejlepším možným způsobem. Předpokládejme, že stroj j potřebuje c_{ij} časových jednotek, aby mohl zpracovat úlohu i . Pokud budeme předpokládat, že tyto stroje pracují sériově, pak chceme minimalizovat lineární součtovou účelovou funkci, a takový problém se označuje jako lineární součtový přiřazovací problém (linear sum assignment problem, LSAP). Pokud budeme předpokládat, že tyto stroje pracují paralelně, pak chceme minimalizovat lineární úzkoprofilovou účelovou funkci, a takový problém se označuje jako lineární úzkoprofilový přiřazovací problém (linear bottleneck assignment problem, LBAP). [14, s. 5]

2.3.1 Lineární součtový přiřazovací problém

LSAP je jedním z nejznámějších problémů lineárního programování a kombinatorické optimalizace. Neformálně řečeno, je dána matice nákladů přiřazení $\mathbf{C} = (c_{ij})$ a chceme vybrat n prvků tak, že v každém řádku a v každém sloupci je vybrán právě jeden prvek a součet odpovídajících nákladů je minimální.

LSAP může být modelován jako minimalizace účelové funkce

$$z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (30)$$

za podmínek

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (31)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (32)$$

kde

- n je počet úloh/strojů,

- c_{ij} je doba zpracování úlohy i na stroji j ,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže úloha i je přiřazena ke stroji j , jinak je její hodnota 0,

dle [14, s. 5].

Podmínky (31) a (32) zajišťují, že každá úloha je přiřazena k jinému stroji.

Dále se předpokládá, že doba c_{ij} zpracování úlohy i na stroji j je nezáporná.

2.3.2 Lineární úzkoprofilový přiřazovací problém

LBAP byl představen Fulkersonem, Glicksbergem a Grossem a vyskytuje se např. v souvislosti s přiřazením úloh k paralelním strojům tak, aby se minimalizoval čas dokončení poslední úlohy. [14, s. 171]

Neformálně řečeno, je dána matice nákladů přiřazení $\mathbf{C} = (c_{ij})$ a chceme vybrat n prvků tak, že v každém řádku a v každém sloupci je vybrán právě jeden prvek a maximální náklad z odpovídajících nákladů je minimální.

LBAP může být modelován jako minimalizace účelové funkce

$$z = \max_{1 \leq i, j \leq n} c_{ij} x_{ij} \quad (33)$$

za podmínek

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (34)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (35)$$

kde

- n je počet úloh/strojů,
- c_{ij} je doba zpracování úlohy i na stroji j ,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže úloha i je přiřazena ke stroji j , jinak je její hodnota 0,

dle [14, s. 172].

Podmínky (34) a (35) zajišťují, že každá úloha je přiřazena k jinému stroji.

2.3.3 k -součtový přiřazovací problém

k -součtový přiřazovací problém (Sum- k assignment problem, Sum- k AP) je variantou LSAP resp. LBAP.

Pro $k = n$ je Sum- k AP ekvivalentní problému LSAP, pro $k = 1$ je Sum- k AP ekvivalentní problému LBAP. [14, s. 195]

Neformálně řečeno, je dána matice nákladů přiřazení $\mathbf{C} = (c_{ij})$ a chceme vybrat n prvků tak, že v každém řádku a v každém sloupci je vybrán právě jeden prvek a součet k maximálních nákladů je minimální. [14, s. 195]

2.3.4 Vyvážený přiřazovací problém

Vyvážený přiřazovací problém (Balanced assignment problem, BAP) se vyskytuje např. v souvislosti s přiřazením úloh ke strojům tak, aby rozdíl mezi maximální a minimální dobou zpracování byl minimální. [14, s. 195]

Neformálně řečeno, je dána matice nákladů přiřazení $\mathbf{C} = (c_{ij})$ a chceme vybrat n prvků tak, že v každém řádku a v každém sloupci je vybrán právě jeden prvek a rozdíl mezi maximálním a minimálním nákladem byl minimální.

BAP může být modelován jako minimalizace účelové funkce

$$z = \max_{1 \leq i, j \leq n} c_{ij} x_{ij} - \min_{1 \leq i, j \leq n} c_{ij} x_{ij} \quad (36)$$

za podmínek

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (37)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (38)$$

kde

- n je počet úloh/strojů,
- c_{ij} je doba zpracování úlohy i na stroji j ,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže úloha i je přiřazena ke stroji j , jinak je její hodnota 0,

dle [14, s. 195].

Podmínky (37) a (38) zajišťují, že každá úloha je přiřazena k jinému stroji.

2.4 Kvadratický přiřazovací problém

Lineární účelové funkce nejsou jediné důležité účelové funkce přiřazovacích problémů. V praxi hrají důležitou roli přiřazovací problémy s kvadratickou účelovou funkcí.

Kvadratický přiřazovací problém (Quadratic assignment problem, QAP) patří mezi nejtěžší problémy kombinatorické optimalizace. QAP byl představen Koopmansem a Beckmannem v roce 1957 jako matematický model pro umístění nedělitelných ekonomických aktivit. Chceme přiřadit n zařízení do n míst s náklady úměrnými součtu násobků toků mezi zařízeními a vzdáleností mezi místy, plus případné náklady na umístění zařízení do zvolených míst. Cílem je přidělit každé zařízení na jiné místo tak, aby celkové náklady byly minimalizovány. [14, s. 203]

QAP může být popsán pomocí tří matic rozměru $n \times n$ v následující formě:

- $\mathbf{A} = (a_{ik})$ je matice toků, kde a_{ik} je tok mezi zařízením i a zařízením k ,
- $\mathbf{B} = (b_{jl})$ je matice vzdáleností, kde b_{jl} je vzdálenost mezi místem j a místem l ,
- $\mathbf{C} = (c_{ij})$ je matice nákladů na přiřazení, kde c_{ij} je cena přiřazení zařízení i do místa j ,

dle [14, s. 203].

QAP se uvádí buď ve tvaru bez lineární části, který se označuje jako QAP(A, B), nebo ve tvaru s lineární částí, který se označuje jako QAP(A, B, C). [14, s. 204]

QAP může být modelován jako minimalizace účelové funkce

$$z = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n a_{ik} b_{jl} x_{ij} x_{kl} \quad (39)$$

v případě QAP(A, B), resp. jako minimalizace účelové funkce

$$z = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sum_{l=1}^n a_{ik} b_{jl} x_{ij} x_{kl} + \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \quad (40)$$

v případě QAP(A, B, C), za podmínek

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (41)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (42)$$

kde

- n je počet zařízení/míst,
- a_{ik} je tok mezi zařízením i a zařízením k ,
- b_{jl} je vzdálenost mezi místem j a místem l ,
- c_{ij} jsou náklady na přiřazení zařízení i do místa j ,
- x_{ij} (x_{kl}) je binární proměnná, jejíž hodnota je 1, jestliže zařízení i je přiřazeno do místa j (zařízení k je přiřazeno do místa l), jinak je její hodnota 0,

dle [14, s. 211].

Podmínky (41) a (42) zajišťují, že každé zařízení je umístěno do jiného místa.

Dále se předpokládá, že tok a_{ik} mezi zařízením i a zařízením k , vzdálenost b_{jl} mezi místem j a místem l a náklady c_{ij} na přiřazení zařízení i do místa j jsou nezáporné.

2.4.1 Kvadratický úzkoprofilový přiřazovací problém

Kvadratický úzkoprofilový přiřazovací problém (Quadratic bottleneck assignment problem, QBAP) získáme nahrazením sumy v účelové funkci QAP maximální hodnotou. Příkladem je situace, kdy chceme přiřadit n zařízení do n míst s náklady úměrnými součtu násobků toků mezi zařízeními a vzdáleností mezi místy. Cílem je přiřadit každé zařízení na jiné místo tak, aby byla minimalizována maximální vzdálenost míst násobená tokem mezi zařízeními. [14, s. 281]

QBAP může být modelován jako minimalizace účelové funkce

$$\max_{1 \leq i, j, k, l \leq n} a_{ik} b_{jl} x_{ij} x_{kl} \quad (43)$$

v případě QBAP(A, B), resp. jako minimalizace účelové funkce

$$\max \left(\max_{1 \leq i, j, k, l \leq n} a_{ik} b_{jl} x_{ij} x_{kl}, \max_{1 \leq i, j \leq n} c_{ij} x_{ij} \right) \quad (44)$$

v případě QBAP(A, B, C), kde

$$\sum_{j=1}^n x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (45)$$

$$\sum_{i=1}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (46)$$

kde

- n je počet zařízení/míst,
- a_{ik} je tok mezi zařízením i a zařízením k ,
- b_{jl} je vzdálenost mezi místem j a místem l ,
- c_{ij} jsou náklady na přiřazení zařízení i do místa j ,
- x_{ij} (x_{kl}) je binární proměnná, jejíž hodnota je 1, jestliže zařízení i je přiřazeno do místa j (zařízení k je přiřazeno do místa l), jinak je její hodnota 0,

dle [14, s. 281].

Podmínky (45) a (46) zajišťují, že každé zařízení je umístěno do jiného místa.

2.4.2 Kvadratický semi-přiřazovací problém

Kvadratický semi-přiřazovací problém (Quadratic semi-assignment problem, semi-QAP), který byl představen Greenbergem, má stejnou účelovou funkci jako QAP, řešením ale nejsou permutace, ale funkce mapující prvky množiny $A = \{1, \dots, n\}$ na prvky množiny $B = \{1, \dots, m\}$, kde $n > m$. Pokud uvažujeme příklad přiřazování zařízení do míst, pak je zde n zařízení, ale pouze m míst a není zde omezení počtu zařízení, které můžeme přiřadit do stejného místa. Podobně jako v případě QAP může být semi-QAP popsán pomocí matice toků $\mathbf{A} = (a_{ik})$ rozměru $n \times n$, matice vzdáleností $\mathbf{B} = (b_{jl})$ rozměru $m \times m$ a matice nákladů na přiřazení $\mathbf{C} = (c_{ij})$ rozměru $n \times m$. [14, s. 293, 294]

semi-QAP může být modelován jako minimalizace účelové funkce

$$\sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^n \sum_{l=1}^m a_{ik} b_{jl} x_{ij} x_{kl} \quad (47)$$

v případě semi-QAP(A, B), resp. jako minimalizace účelové funkce

$$\sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^n \sum_{l=1}^m a_{ik} b_{jl} x_{ij} x_{kl} + \sum_{i=1}^n \sum_{j=1}^m c_{ij} x_{ij} \quad (48)$$

v případě semi-QAP(A, B, C), za podmínky

$$\sum_{j=1}^m x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (49)$$

kde

- n je počet zařízení/míst,
- a_{ik} je tok mezi zařízením i a zařízením k ,
- b_{jl} je vzdálenost mezi místem j a místem l ,
- c_{ij} jsou náklady na přiřazení zařízení i do místa j ,
- x_{ij} (x_{kl}) je binární proměnná, jejíž hodnota je 1, jestliže zařízení i je přiřazeno do místa j (zařízení k je přiřazeno do místa l), jinak je její hodnota 0,

dle [14, s. 294].

Podmínka (49) zajišťuje, že každé zařízení je umístěno pouze do jednoho místa.

2.5 Problém rozvrhování proudové výroby

Problém rozvrhování proudové výroby (Flow shop scheduling problem, FSSP) patří do oblasti problémů rozvrhování (scheduling problems), přesněji do oblasti problémů rozvrhování výroby (shop scheduling problems).

V mnoha výrobních a montážních zařízeních musí každá úloha podstoupit řadu operací. Tyto operace musí být často provedeny u všech úloh ve stejném pořadí, což znamená, že úlohy se musí řídit stejnou cestou. Stroje jsou pak předpokládány v sériovém uspořádání a prostředí se označuje jako proudová výroba (flow shop). [15, s. 151]

Flow shop Fm , který je základní reprezentací problému rozvrhování proudové výroby, je formulován následovně: Je dáno m strojů v sérii a n úloh. Každá úloha musí být zpracována na každém stroji. Všechny úlohy musí postupovat stejnou cestou, tj. musí být zpracovány nejprve na stroji 1, pak na stroji 2 atd. Po dokončení úlohy na stroji se úloha přesune do fronty následujícího stroje. Obvykle jsou všechny fronty předpokládány jako fronty s disciplínou FIFO (First in first out), což znamená, že úloha přicházející do fronty jako první, je také zpracovávána jako první. [15, s. 152]

Problém rozvrhování je popsán trojicí $\alpha | \beta | \gamma$, kde α udává strojové prostředí (Fm v případě flow shop), β udává omezení a γ udává účelovou funkci, která má být minimalizována. [15, s. 14]

Mezi základní varianty FSSP patří problém rozvrhování proudové výroby s neomezeným skladem (flow shop scheduling problem with unlimited intermediate storage, FSSPUIS), problém rozvrhování proudové výroby s omezeným skladem (flow shop scheduling problem with limited intermediate storage, FSSPLIS) a problém rozvrhování proudové výroby s nulovým zpožděním (flow shop scheduling problem with no-wait, FSSPNW). [15, s. 17]

V následujících variantách FSSP bude jako účelová funkce uvažována minimalizace celkového času dokončení, tzv. makespan (C_{max}), který je definován jako

$$C_{max} = \max(C_i) \quad i = 1, 2, \dots, n$$

kde

- n je počet úloh,
- C_i je doba dokončení úlohy i ,

dle [15, s. 18].

2.5.1 Problém rozvrhování proudové výroby s neomezeným skladem

FSSPUIS je případ FSSP, kde skladovací kapacita mezi po sobě jdoucími stroji je neomezená. Po dokončení úlohy na stroji se tato úloha zařadí do fronty následujícího stroje, kde čeká na zpracování. Pokud uvažujeme fronty s disciplínou FIFO, pak jde o permutační problém rozvrhování proudové výroby. Pokud jako cíl uvažujeme minimalizaci celkového času dokončení, lze problém zapsat ve tvaru

$$Fm | pmu | C_{max}$$

kde pmu udává, že jde o permutační problém, a C_{max} udává, že cílem je minimalizace času dokončení poslední úlohy na posledním stroji. [16, s. 52]

Předpokládejme permutační rozvrh pro zpracování n úloh v proudové výrobě tvořené m stroji. Doba zpracování každého úkolu j na stroji i je dána jako p_{ij} . Doba dokončení C_{ij} úlohy j na stroji i může být vypočtena pomocí souboru rekurzních rovnic

$$C_{i1} = \sum_{k=1}^i p_{k1} \quad i = 1, 2, \dots, m \quad (50)$$

$$C_{ij} = \sum_{k=1}^j p_{ik} \quad j = 1, 2, \dots, n \quad (51)$$

$$C_{ij} = \max(C_{i-1j}, C_{ij-1}) + p_{ij} \quad i = 2, 3, \dots, m \quad j = 2, 3, \dots, n \quad (52)$$

kde

- n je počet úloh,
- m je počet strojů,
- p_{ij} je doba zpracování úlohy j na stroji i ,
- C_{ij} je doba dokončení úlohy j na stroji i ,

dle [15, s. 152].

2.5.2 Problém rozvrhování proudové výroby s omezeným skladem

FSSPLIS je případ FSSP, kde skladovací kapacita mezi po sobě jdoucími stroji je omezená. Každý problém rozvrhování proudové výroby s omezeným (ale konečným) skladem lze modelovat jako problém rozvrhování proudové výroby s nulovým skladem, proto bude dále uvažován pouze tento případ. Když stroj dokončí zpracovávání úlohy, úloha nemůže být přemístěna do následujícího stroje, pokud je tento stroj zaneprázdněn. Úloha musí zůstat na aktuálním stroji, který pak nemůže začít zpracovávat následující úlohu. Tento jev se označuje jako blokování. Problém rozvrhování proudové výroby s nulovým skladem, kde je cílem minimalizace celkového času dokončení, lze zapsat ve tvaru

$$Fm \mid block \mid C_{max}$$

dle [15, s. 163, 164].

Předpokládejme permutační rozvrh pro zpracování n úloh v proudové výrobě tvořené m stroji. Doba zpracování každého úkolu j na stroji i je dána jako p_{ij} . Doba odchodu D_{ij} úlohy j ze stroje i může být vypočtena pomocí souboru rekurzíčních rovnic

$$D_{i1} = \sum_{k=1}^i p_{k1} \quad i = 1, 2, \dots, m \quad (53)$$

$$D_{ij} = \max(D_{i-1,j} + p_{ij}, D_{i+1,j-1}) \quad i = 1, 2, \dots, m-1 \quad j = 2, 3, \dots, n \quad (54)$$

$$D_{m,j} = D_{m-1,j} + p_{mj} \quad j = 2, 3, \dots, n \quad (55)$$

kde

- n je počet úloh,
- m je počet strojů,
- p_{ij} je doba zpracování úlohy j na stroji i ,
- D_{ij} je doba, kdy úloha j odchází ze stroje i ,

dle [15, s. 164].

Je zřejmé, že mezi dobou odchodu D_{ij} úlohy j ze stroje i a dobou dokončení C_{ij} úlohy j na stroji i platí $D_{ij} \geq C_{ij}$. Rovnost platí pro případ, kdy úloha j není blokována. Doba zahájení zpracování úlohy j na prvním stroji se označuje D_{0j} a platí, že

$$D_{01} = 0 \quad (56)$$

$$D_{0j} = D_{1,j-1} \quad j = 2, 3, \dots, n \quad (57)$$

dle [15, s. 164].

2.5.3 Problém rozvrhování proudové výroby s nulovým zpožděním

FSSPNW je případ FSSP, kde skladovací kapacita mezi po sobě jdoucími stroji je nulová a úloha procházející systémem nesmí čekat na žádném stroji. To znamená, že vždy, když stroj dokončí zpracování úlohy, následující stroj musí být nečinný, aby úloha nemusela čekat. Problém rozvrhování proudové výroby s nulovým zpožděním, kde je cílem minimalizace celkového času dokončení, lze zapsat ve tvaru

$$Fm \mid nwt \mid C_{max}$$

dle [15, s. 170].

Předpokládejme permutační rozvrh pro zpracování n úloh v proudové výrobě tvořené m stroji. Doba zpracování každého úkolu j na stroji i je dána jako p_{ij} . Každý úkol j je tvořen sekvencí m operací. Pro zajištění nulového zpoždění musí být doba dokončení operace rovna době začátku zpracování následující operace. Jinými slovy, mezi dvěma po sobě jdoucími operacemi každé úlohy nesmí být žádné zpoždění. [16, s. 72, 73]

Minimální zpoždění d_{j-1j} mezi začátkem zpracování úlohy $j - 1$ a začátkem zpracování úlohy j na prvním stroji lze v problému rozvrhování proudové výroby s nulovým zpožděním vypočítat jako

$$d_{j-1j} = p_{1j-1} + \max \left[\max_{2 \leq i \leq m} \left(\sum_{k=2}^i p_{kj-1} - \sum_{k=1}^{i-1} p_{kj} \right), 0 \right] \quad j = 2, 3, \dots, n \quad (58)$$

kde

- m je počet strojů,
- p_{ij} je doba zpracování úlohy j na stroji i ,

dle [16, s. 72, 73].

Z výše uvedeného vyplývá následující: Jestliže minimální zpoždění d_{j-1j} mezi začátkem zpracování úlohy $j - 1$ a začátkem zpracování úlohy j na prvním stroji je rovno p_{1j-1} , tedy době zpracování úlohy $j - 1$ na prvním stroji, pak zpracování úlohy j na prvním stroji může nastat ihned po dokončení úlohy $j - 1$ na prvním stroji, při splnění požadavku na nulové zpoždění.

Dobu dokončení úlohy j je možné vypočítat jako

$$C_1 = \sum_{i=1}^m p_{i1} \quad (59)$$

$$C_j = \sum_{k=2}^j d_{k-1k} + \sum_{i=1}^m p_{ij} \quad j = 2, 3, \dots, n \quad (60)$$

kde

- n je počet úloh,
- m je počet strojů,
- p_{ij} je doba zpracování úlohy j na stroji i ,
- d_{j-1j} je minimální zpoždění mezi začátkem zpracování úlohy $j - 1$ a začátkem zpracování úlohy j na prvním stroji,

dle [16, s. 73].

2.6 Problém obchodního cestujícího

Problém obchodního cestujícího (Travelling salesman problem, TSP) patří asi mezi nejznámější problémy kombinatorické optimalizace. TSP řeší otázku nalezení nejkratší (nejlevnější) cesty mezi n městy, která začíná ve výchozím městě, každé město navštíví právě jednou a vrací se zpět do výchozího města. [17, s. 1]

Základní rozdělení TSP je na symetrický problém obchodního cestujícího (symmetric travelling salesman problem, STSP), asymetrický problém obchodního cestujícího (asymmetric travelling salesman problem, ATSP) a vícenásobný problém obchodního cestujícího (multiple travelling salesman problem, MTSP). [18, s. 1]

2.6.1 Symetrický problém obchodního cestujícího

STSP je nejjednodušší variantou TSP, kdy vzdálenost resp. cena cesty mezi dvěma městy je stejná v obou směrech, tzn. cena cesty d_{ij} z města i do města j je stejná jako cena cesty d_{ji} z města j do města i . [18, s. 1]

STSP může být modelován jako minimalizace účelové funkce

$$z = \sum_{i=1}^n \sum_{\substack{j=1 \\ i \neq j}}^n d_{ij} x_{ij} \quad (61)$$

za podmínek

$$\sum_{\substack{j=1 \\ i \neq j}}^n x_{ij} = 1 \quad i = 1, 2, \dots, n \quad (62)$$

$$\sum_{\substack{i=1 \\ i \neq j}}^n x_{ij} = 1 \quad j = 1, 2, \dots, n \quad (63)$$

kde

- n je počet měst,
- d_{ij} je vzdálenost mezi městem i a městem j ,
- x_{ij} je binární proměnná, jejíž hodnota je 1, jestliže obchodní cestující jede z města i do města j , jinak je její hodnota 0,

dle [19].

Podmínky (62) a (63) zajišťují, že každé město je navštíveno právě jednou.

Dále se předpokládá, že pro vzdálenost mezi městem i a městem j platí $d_{ij} > 0$.

2.6.2 Asymetrický problém obchodního cestujícího

Jak již bylo uvedeno, v případě STSP je vzdálenost resp. cena cesty mezi dvěma městy stejná v obou směrech. Toto však neplatí v případě ATSP, jelikož cesta nemusí existovat v obou směrech nebo vzdálenost v obou směrech nemusí být stejná. Příkladem mohou být dopravní kolize, jednosměrné ulice, cena letenky pro města s různými příletovými a odletovými poplatky apod. Obecně platí, jestliže $d_{ij} \neq d_{ji}$ pro alespoň jeden pár měst i a j , TSP se stává ATSP. [18, s. 2]

2.6.3 Vícenásobný problém obchodního cestujícího

V případě MTSP se vyskytuje m obchodních cestujících, kteří se nacházejí ve výchozím městě. Úkolem na nalezení cest pro obchodní cestující takových, že každé město je navštíveno právě jednou a celková vzdálenost resp. cena cest všech obchodních cestujících je minimalizována. Jestliže je počet obchodních cestujících $m = 1$, pak se MTSP stává TSP. [18, s. 2]

2.7 Kapacitní rozvozní problém

Rozvozní problém (Vehicle routing problem, VRP) řeší otázku stanovení optimální množiny cest, které mají být provedeny množinou vozidel, s cílem obsloužit množinu zákazníků. Tento problém byl poprvé představen v roce 1959 Dantzigem a Ramserem. [20, s. 1]

Základní a nejvíce studovanou verzí VRP je kapacitní rozvozní problém (capacitated vehicle routing problem, CVRP). Z CVRP následně vycházejí varianty, mezi nimiž nejznámější jsou: vzdálenostně omezený rozvozní problém (distance-constrained vehicle routing problem, DVRP), rozvozní problém s časovými okny (vehicle routing problem with time windows, VRPTW), rozvozní problém se zpětným svozem (vehicle routing problem with backhauls, VRPB) a rozvozní problém s vyzvednutím a doručením (vehicle routing problem with pickup and delivery, VRPPD). [20, s. 5]

CVRP je tvořen zákazníky s požadavky a souborem vozidel se shodnou kapacitou. Všechna vozidla začínají a končí svoji cestu v depu, jež má fiktivní nulový požadavek. Každé vozidlo zajišťuje obsluhu podmnožiny zákazníků a může vykonat nejvýše jednu cestu. Každý zákazník je obsloužen právě jedním vozidlem. Cílem je obsloužit všechny zákazní-

ky tak, aby nebyla překročena kapacita žádného vozidla a celková vzdálenost ujetá všemi vozidly byla minimalizována. [20, s. 5, 6]

CVRP může být modelován jako minimalizace účelové funkce

$$z = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^m dist_{ij} x_{ijk} \quad (64)$$

za podmínek

$$\sum_{i=1}^n d_i y_{ik} \leq c \quad k = 1, 2, \dots, m \quad (65)$$

$$\sum_{k=1}^m y_{ik} = 1 \quad i = 1, 2, \dots, n \quad (66)$$

$$\sum_{i=1}^n x_{ijk} = y_{kj} \quad j, k = 1, 2, \dots, m \quad (67)$$

$$\sum_{i=1}^n x_{ijk} = y_{ki} \quad i = 1, 2, \dots, n \quad k = 1, 2, \dots, m \quad (68)$$

kde

- n je počet zákazníků,
- m je počet vozidel,
- d_i je požadavek zákazníka i ,
- $dist_{ij}$ je vzdálenost mezi zákazníkem i a zákazníkem j ,
- c je kapacita vozidla,
- y_{ik} je binární proměnná, jejíž hodnota je 1, jestliže zákazník i je obsloužen vozidlem k , jinak je její hodnota 0,
- x_{ijk} je binární proměnná, jejíž hodnota je 1, jestliže vozidlo k jede od zákazníka i k zákazníkovi j , jinak je její hodnota 0,

dle [21, s. 36, 37].

Rovnice (64) je účelovou funkcí CVRP, podmínka (65) zajišťuje, aby nebyla překročena kapacita vozidla, podmínka (66) zajišťuje, že každý požadavek zákazníka je obsloužen

právě jedním vozidlem, podmínky (67) a (68) zajišťují, že právě jedno vozidlo přijíždí k zákazníkovi a odjíždí od zákazníka.

Dále se předpokládá, že

- požadavek d_i každého zákazníka i a kapacita vozidla c jsou kladné hodnoty

$$d_i, c > 0 \quad i = 1, 2, \dots, n \quad (69)$$

- požadavek d_i každého zákazníka i je menší nebo roven kapacitě vozidla c

$$d_i \leq c \quad i = 1, 2, \dots, n \quad (70)$$

dle [21, s. 36].

2.7.1 Vzdálenostně omezený rozvozní problém

DVRP je variantou CVRP, kde kapacitní omezení je nahrazeno omezením vzdálenosti, které je kladeno na maximální délku cesty. V případě, kdy je uvažováno jak vzdálenostní i kapacitní omezení, je problém označován jako vzdálenostně omezený kapacitní rozvozní problém (distance-constrained capacitated vehicle routing problem, DCVRP). [20, s. 8]

2.7.2 Rozvozní problém s časovými okny

VRPTW je rozšířením CVRP, kde každému zákazníkovi je přiřazena doba obsluhy a časový interval obsluhy, který se označuje jako časové okno (time window). Obsluha zákazníka musí začít v odpovídajícím časovém okně a vozidlo musí zůstat u zákazníka po odpovídající dobu obsluhy. V případě brzkého příjezdu vozidla k zákazníkovi je vozidlu dovoleno čekat do doby zahájení obsluhy. [20, s. 157]

2.7.3 Rozvozní problém se zpětným svozem

VRPB je rozšířením CVRP, kde množina zákazníků je rozdělena na dvě podmnožiny. První podmnožina obsahuje zákazníky (tzv. linehaul customers), kteří požadují doručení určitého požadavku, druhá podmnožina obsahuje zákazníky (tzv. backhaul customers), kteří požadují vyzvednutí určitého požadavku. VRPB uvažuje následující přednostní omezení: V případě, kdy je cesta tvořena oběma typy zákazníků, pak linehaul customers jsou obslouženi před backhaul customers. [20, s. 195]

2.7.4 Rozvozní problém s vyzvednutím a doručením

VRPPD je rozšířením CVRP, kde každý zákazník má požadavek jak na doručení, tak na vyzvednutí. U každého zákazníka je kromě samotných požadavků na doručení a vyzvednutí specifikováno, kam má být doručen u zákazníka vyzvednutý požadavek a kde se nachází požadavek k doručení zákazníkovi. Tato místa jsou obvykle uvažována jako depo. Dále se předpokládá, že doručení je pro každého zákazníka provedeno před vyzvednutím. [20, s. 225]

II. PRAKTICKÁ ČÁST

3 NÁVRH KNIHOVNY OPERAČNÍHO VÝZKUMU

Tato kapitola se zabývá návrhem knihovny operačního výzkumu (dále jen knihovny). Knihovna bude implementovat vybrané problémy operačního výzkumu (dále jen problémy), jejichž popis je součástí teoretické části.

Jako programové prostředí pro realizaci knihovny byl zvolen objektově orientovaný jazyk C++.

3.1 Požadavky na knihovnu operačního výzkumu

V této části je uveden přehled požadavků, které jsou kladeny na knihovnu. Základním požadavkem kladeným na knihovnu je modularita, tj. možnost snadného rozšíření o nový problém resp. variantu již implementovaného problému. U jednotlivých problémů knihovny musí být umožněno

- nastavení dat problému,
- generování dat problému,
- import dat problému ze souboru,
- export dat problému do souboru,
- zobrazení dat problému,
- výpočet hodnoty účelové funkce problému,
- export výsledků.

Nastavení dat problému

Nastavením dat problému se rozumí možnost nastavení konkrétních dat, které však musí respektovat všechna omezení, která jsou na data problému kladena. Příkladem je možnost nastavení dat problému batohu na konkrétní kapacitu batohu c a počet předmětů n , kde každý předmět i je specifikován hmotností w_i a cenou p_i .

Generování dat problému

Generováním dat problému se rozumí možnost generování dat dle určitých požadavků, které však musí respektovat všechna omezení, která jsou na data problému kladena. Příkladem je možnost vygenerovat data pro problém rozvrhování proudové výroby, kde je třeba

zpracovat n úloh na m strojích, kde doba zpracování úlohy i na stroji j se nachází v časovém intervalu $\langle a; b \rangle$.

Import/export dat problému ze/do souboru

Importem dat ze souboru se rozumí možnost načtení existujících dat problému ze souboru daného formátu. Exportem dat do souboru se rozumí možnost uložení aktuálních dat problému do souboru daného formátu. Pro možnost importu/exportu dat je nutné navrhnout strukturu těchto souborů.

Zobrazení dat problému

Zobrazením dat problému se rozumí možnost zobrazení aktuálně používaných/načtených dat problému.

Výpočet hodnoty účelové funkce problému

Výpočtem hodnoty účelové funkce problému se rozumí možnost ohodnocení dat problému pro zvolený plán na základě účelové funkce problému.

Export výsledků

Exportem výsledků se rozumí možnost uložení plánu a odpovídající hodnoty účelové funkce do souboru daného formátu.

3.2 Návrh základní struktury knihovny operačního výzkumu

Na základě požadavků, které jsou kladeny na knihovnu, byl proveden návrh základní struktury knihovny.

3.2.1 Rozhraní *IData*

Pro import/export dat problému ze/do souboru a zobrazení dat problému bylo navrženo rozhraní *IData*, které poskytuje čistě virtuální metody

- *virtual void readDataFromFile(const std::string& filename) = 0* - import dat problému ze souboru,
- *virtual void writeDataToFile(const std::string& filename) = 0* - export dat problému do souboru,
- *virtual void showData() = 0* - zobrazení dat problému.

Tyto čistě virtuální metody rozhraní je třeba implementovat v odvozené třídě.

3.2.2 Třída obecného problému

Jako obecný problém knihovny lze uvažovat třídu *Problem* odvozenou od rozhraní *IData*. Tato třída obecného problému zapouzdřuje proměnné problému, implementuje čistě virtuální metody rozhraní *IData* a dále poskytuje metody

- *Problem::Problem* - konstruktor ve variantách pro nastavení, generování a import dat problému,
- *Problem::setData* - nastavení dat problému,
- *Problem::generateData* - generování dat problému dle požadavků,
- *Problem::evaluateXYZ* - výpočet hodnoty účelové funkce problému (varianty *XYZ* problému) a přetížená varianta doplněná o export výsledků.

Parametry a návratové hodnoty metod obecného problému *Problem* nejsou záměrně uvedeny, jelikož jsou závislé na konkrétním problému knihovny. Je vhodné poznamenat, že třída obecného problému *Problem* je zmíněna pouze pro demonstraci návrhu struktury knihovny.

3.2.3 Pomocné třídy

Dále byly navrženy pomocné třídy pro generování pseudonáhodných hodnot v zadaném intervalu, pro práci se souřadnicemi a pro generování/ukládání/načítání plánu.

Třída *RandomHelper*

Pro generování pseudonáhodných hodnot v zadaném intervalu byla navržena pomocná třída *RandomHelper*, která poskytuje metody

- *RandomHelper::RandomHelper()* - konstruktor - náhodná inicializace (seed) generátoru,
- *RandomHelper::RandomHelper(unsigned int seed)* - konstruktor - konkrétní inicializace (seed) generátoru,
- *double RandomHelper::getRandomDoubleInRange(double min, double max)* - generování reálné hodnoty v intervalu $\langle min; max \rangle$,
- *int RandomHelper::getRandomIntegerInRange(int min, int max)* - generování celočíselné hodnoty v intervalu $\langle min; max \rangle$.

Třída *CoordinatesHelper*

Pro práci se souřadnicemi byla navržena pomocná třída *CoordinatesHelper*, která zapouzdřuje souřadnice x a y a poskytuje metody

- *CoordinatesHelper::CoordinatesHelper(double x, double y)* - konstruktor,
- *double CoordinatesHelper::distance(CoordinatesHelper other)* - vzdálenost mezi aktuálními a jinými souřadnicemi,
- *double CoordinatesHelper::getX() const* - souřadnice x ,
- *double CoordinatesHelper::getY() const* - souřadnice y .

Třída *ScheduleHelper*

Pro generování/ukládání/načítání plánu byla navržena pomocná třída *ScheduleHelper*, která poskytuje metody

- *std::vector<unsigned int> ScheduleHelper::generatePermutationSchedule(unsigned int n)* - generování plánu (permutace),
- *std::vector<unsigned int> ScheduleHelper::generateSelectionSchedule(unsigned int n)* - generování plánu (výběr 1 až n prvků z permutace)
- *std::vector<unsigned int> ScheduleHelper::readScheduleFromFile(const std::string& filename)* - načtení plánu ze souboru,
- *void ScheduleHelper::writeScheduleToFile(const std::vector<unsigned int>& schedule, const std::string& filename)* - zápis plánu do souboru.

3.3 Návrh tříd problémů knihovny operačního výzkumu

Návrh tříd problémů knihovny vychází z návrhu obecného problému. Jednotlivé třídy zapouzdřují proměnné odpovídající danému problému a poskytují metody pro práci s těmito proměnnými. Pro každou variantu problému je dostupná vlastní metoda *evaluateXYZ*, která slouží pro stanovení hodnoty účelové funkce dané varianty problému.

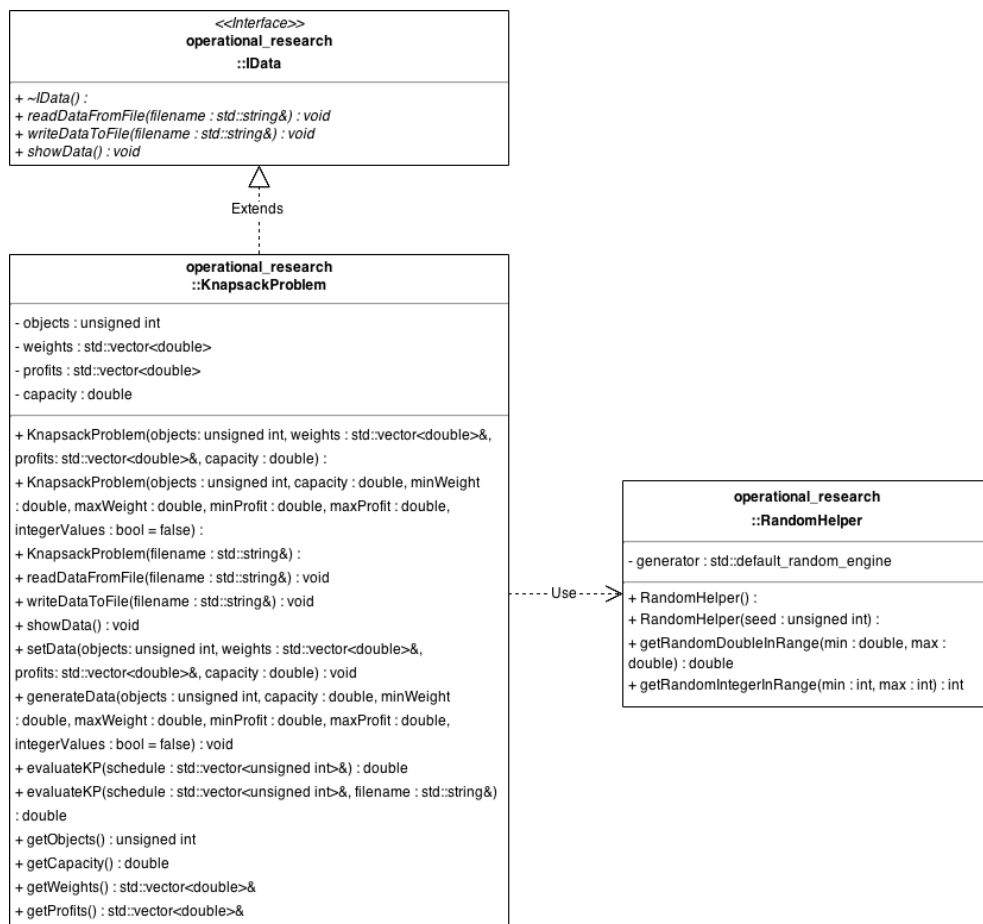
3.3.1 Třída *KnapsackProblem*

Pro problém batohu, jehož popis je součástí kapitoly 2.1, byla navržena třída *KnapsackProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocnou třídu *RandomHelper*.

Třída *KnapsackProblem* zapouzdřuje proměnné

- *unsigned int objects* - celočíselná proměnná udávající počet objektů,
- *std::vector<double> weights* - pole reálných hodnot velikosti [*objects*] udávající hmotnosti objektů, kde hodnota na *i*-té pozici odpovídá hmotnosti *i*-tého objektu,
- *std::vector<double> profits* - pole reálných hodnot velikosti [*objects*] udávající ceny objektů, kde hodnota na *i*-té pozici odpovídá ceně *i*-tého objektu,
- *double capacity* - reálná proměnná udávající kapacitu batohu.

Třída *KnapsackProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení základní varianty problému slouží metoda *KnapsackProblem::evaluateKP*. Diagram třídy *KnapsackProblem* je uveden na obrázku (Obr. 3).



Obr. 3. Diagram třídy *KnapsackProblem*

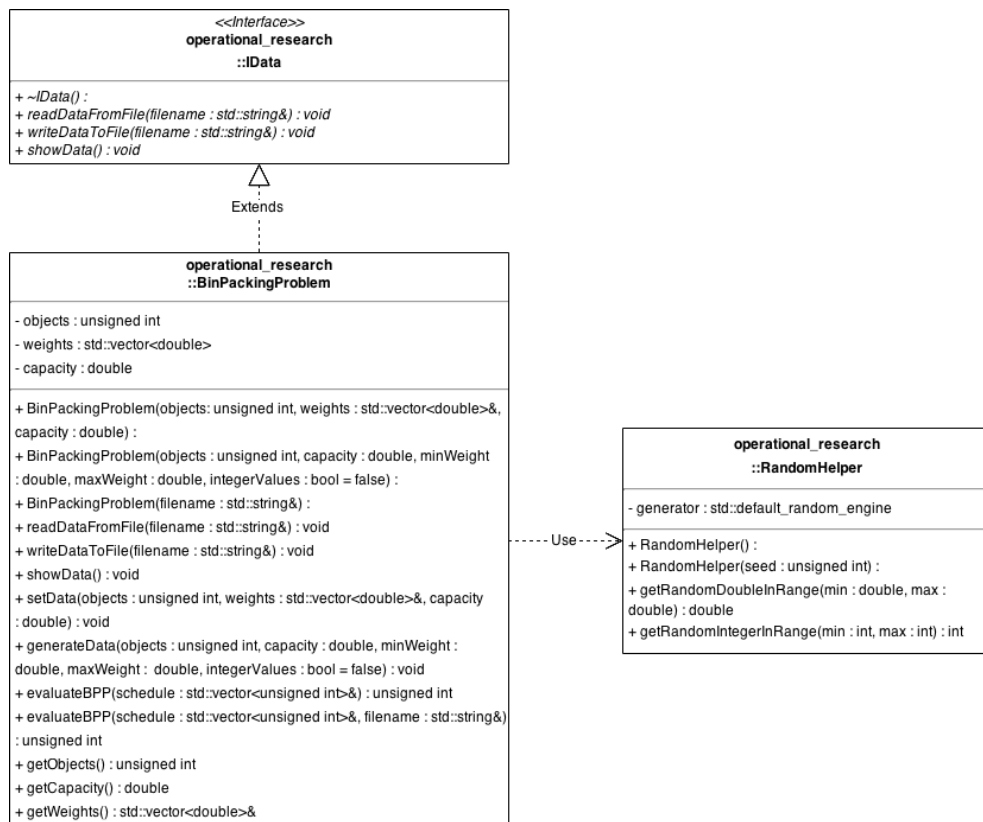
3.3.2 Třída *BinPackingProblem*

Pro problém naplňování zásobníku, jehož popis je součástí kapitoly 2.2, byla navržena třída *BinPackingProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocnou třídu *RandomHelper*.

Třída *BinPackingProblem* zapouzdřuje proměnné

- *unsigned int objects* - celočíselná proměnná udávající počet objektů,
- *std::vector<double> weights* - pole reálných hodnot velikosti [*objects*] udávající hmotnosti objektů, kde hodnota na *i*-té pozici odpovídá hmotnosti *i*-tého objektu,
- *double capacity* - reálná proměnná udávající kapacitu zásobníku.

Třída *BinPackingProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení základní varianty problému slouží metoda *BinPackingProblem::evaluateBPP*. Diagram třídy *BinPackingProblem* je uveden na obrázku (Obr. 4).



Obr. 4. Diagram třídy *BinPackingProblem*

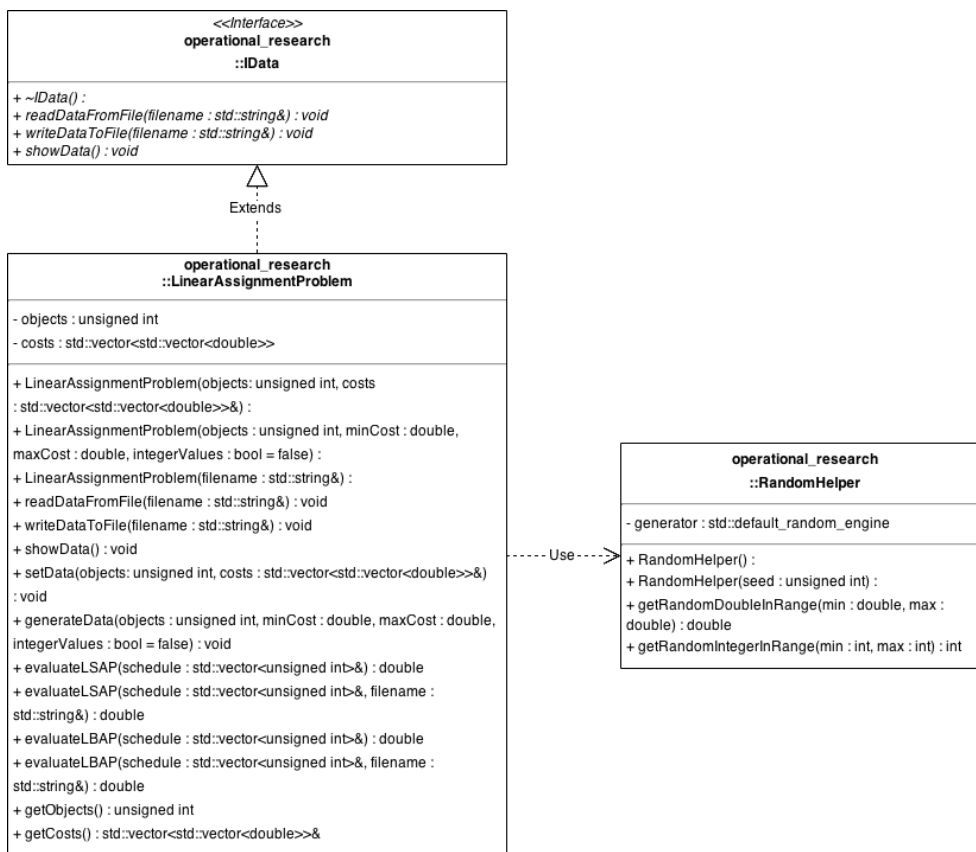
3.3.3 Třída *LinearAssignmentProblem*

Pro lineární přiřazovací problém, jehož popis je součástí kapitoly 2.3, byla navržena třída *LinearAssignmentProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocnou třídu *RandomHelper*.

Třída *LinearAssignmentProblem* zapouzdřuje proměnné

- *unsigned int objects* - celočíselná proměnná udávající počet objektů (např. úlohy a stroje),
- *std::vector<std::vector<double>> costs* - matice reálných hodnot rozměru [*objects* × *objects*] udávající ceny přiřazení, kde hodnota na *i*-tém řádku a *j*-tém sloupci odpovídá ceně přiřazení *i*-té úlohy k *j*-tému stroji.

Třída *LinearAssignmentProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení slouží metody *LinearAssignmentProblem::evaluateLSAP* (lineární součtový přiřazovací problém) a *LinearAssignmentProblem::evaluateLBAP* (lineární úzkoprofilový přiřazovací problém). Diagram třídy *LinearAssignmentProblem* je uveden na obrázku (Obr. 5).



Obr. 5. Diagram třídy *LinearAssignmentProblem*

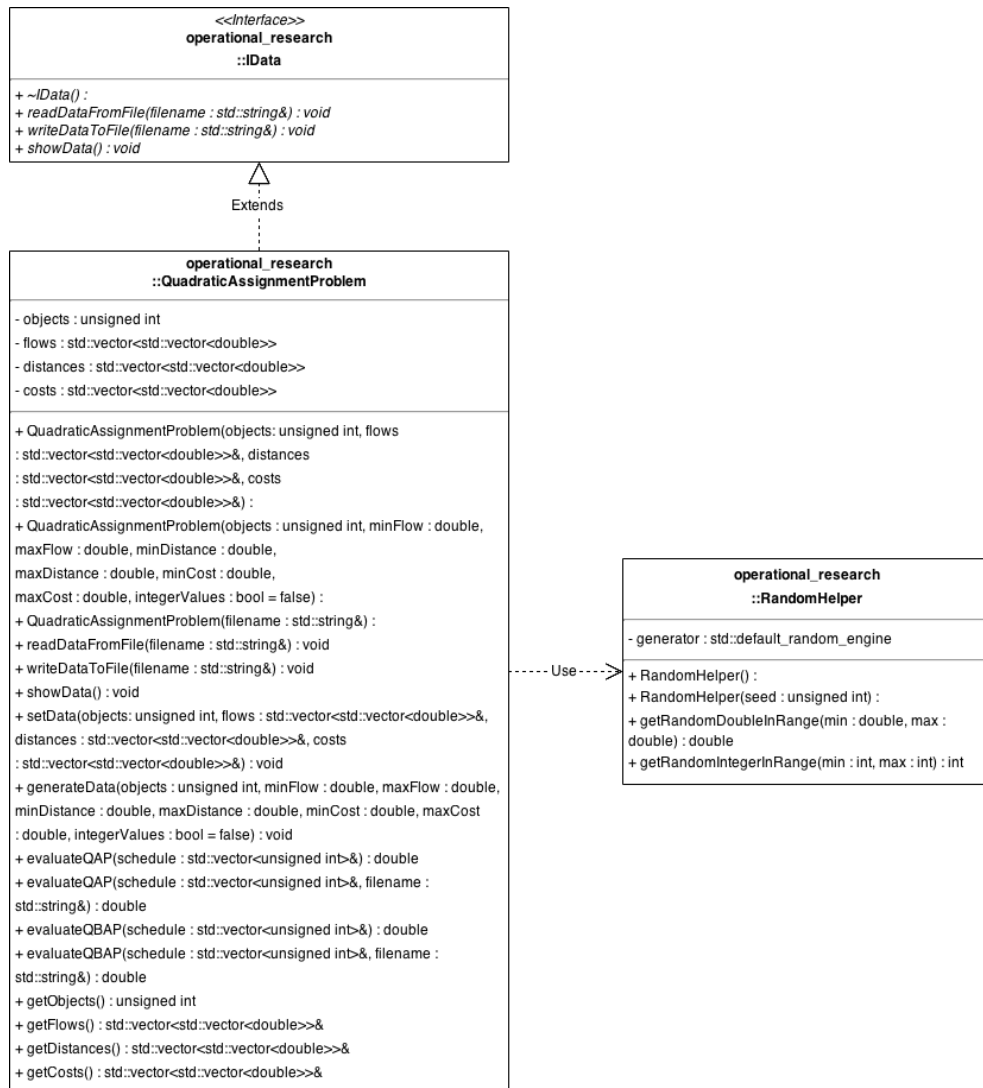
3.3.4 Třída *QuadraticAssignmentProblem*

Pro kvadratický přiřazovací problém, jehož popis je součástí kapitoly 2.4, byla navržena třída *QuadraticAssignmentProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocnou třídu *RandomHelper*.

Třída *QuadraticAssignmentProblem* zapouzdřuje proměnné

- *unsigned int objects* - celočíselná proměnná udávající počet objektů (např. zařízení a místa),
- *std::vector<std::vector<double>> flows* - matice reálných hodnot rozměru [*objects* × *objects*] udávající toky, kde hodnota na *i*-tém řádku a *j*-tém sloupci odpovídá toku mezi *i*-tým zařízením a *j*-tým zařízením,
- *std::vector<std::vector<double>> distances* - matice reálných hodnot rozměru [*objects* × *objects*] udávající vzdálenosti, kde hodnota na *i*-tém řádku a *j*-tém sloupci odpovídá vzdálenosti *i*-tého místa a *j*-tého místa,
- *std::vector<std::vector<double>> costs* - matice reálných hodnot rozměru [*objects* × *objects*] udávající ceny přiřazení, kde hodnota na *i*-tém řádku a *j*-tém sloupci odpovídá ceně přiřazení *i*-tého zařízení do *j*-tého místa.

Třída *QuadraticAssignmentProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení slouží metody *QuadraticAssignmentProblem::evaluateQAP* (kvadratický přiřazovací problém) a *QuadraticAssignmentProblem::evaluateQBAP* (kvadratický úzkoprofilový přiřazovací problém). Diagram třídy *QuadraticAssignmentProblem* je uveden na obrázku (Obr. 6).

Obr. 6. Diagram třídy *QuadraticAssignmentProblem*

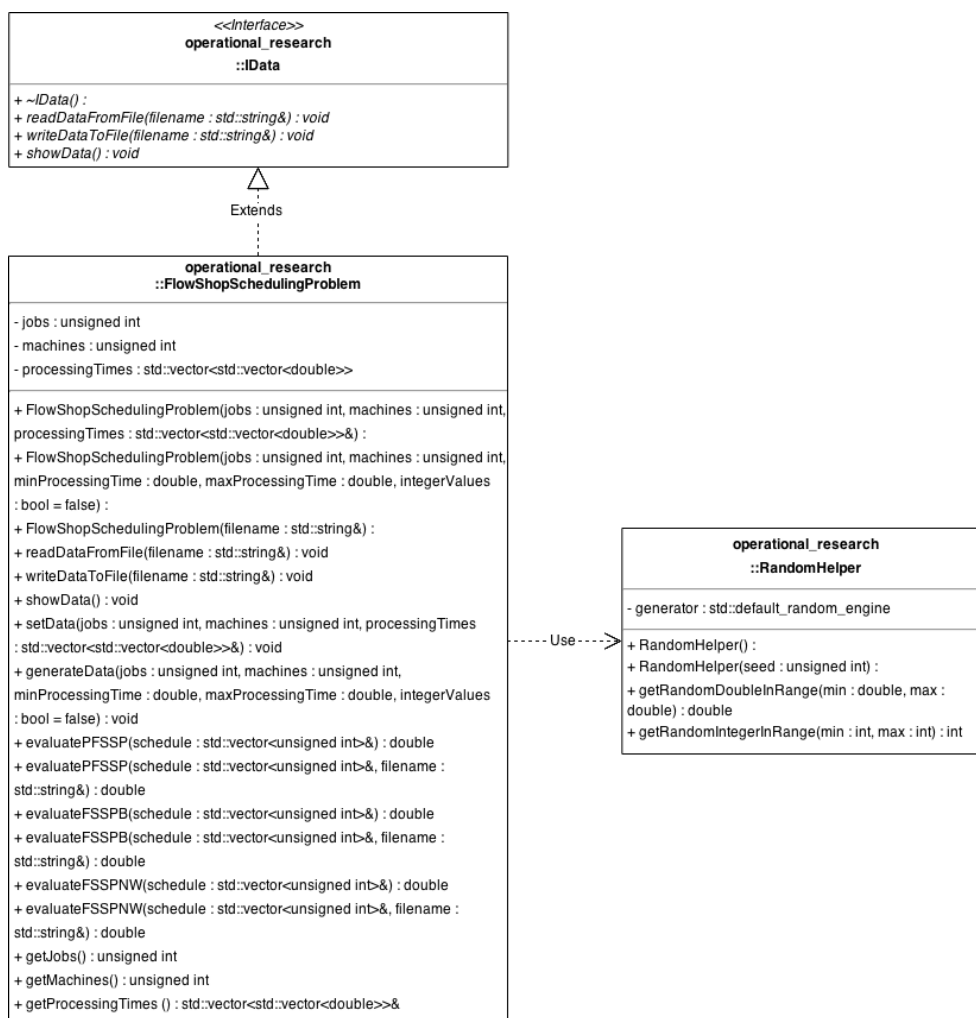
3.3.5 Třída *FlowShopSchedulingProblem*

Pro problém rozvrhování proudové výroby, jehož popis je součástí kapitoly 2.5, byla navržena třída *FlowShopSchedulingProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocnou třídu *RandomHelper*.

Třída *FlowShopSchedulingProblem* zapouzdřuje proměnné

- *unsigned int jobs* - celočíselná proměnná udávající počet úloh,
- *unsigned int machines* - celočíselná proměnná udávající počet strojů,
- *std::vector<std::vector<double>>> processingTimes* - matice reálných hodnot rozměru $[machines \times jobs]$ udávající doby zpracování úloh na strojích, kde hodnota na i -tém řádku a j -tém sloupci odpovídá době zpracování j -té úlohy na i -tém stroji.

Třída *FlowShopSchedulingProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení slouží metody *FlowShopSchedulingProblem::evaluatePFSSP* (permutační problém rozvrhování proudové výroby), *FlowShopSchedulingProblem::evaluateFSSPB* (problém rozvrhování proudové výroby s blokováním) a *FlowShopSchedulingProblem::evaluateFSSPNW* (problém rozvrhování proudové výroby s nulovým zpožděním). Diagram třídy *FlowShopSchedulingProblem* je uveden na obrázku (Obr. 7).



Obr. 7. Diagram třídy *FlowShopSchedulingProblem*

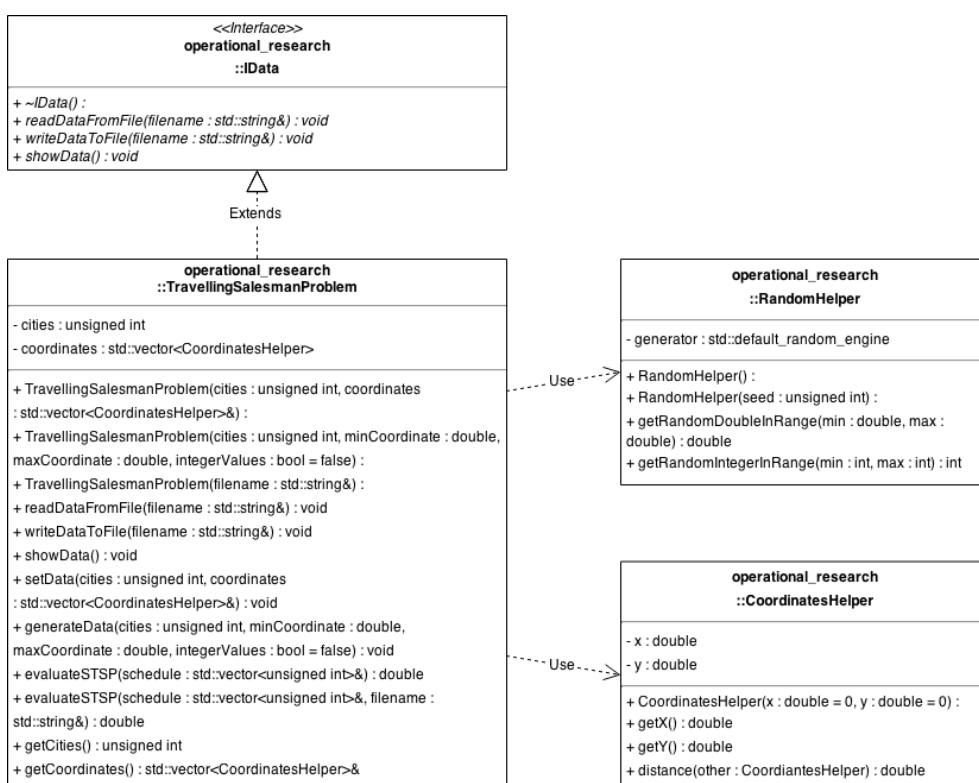
3.3.6 Třída *TravellingSalesmanProblem*

Pro problém obchodního cestujícího, jehož popis je součástí kapitoly 2.6, byla navržena třída *TravellingSalesmanProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocné třídy *RandomHelper* a *CoordinatesHelper*.

Třída *TravellingSalesmanProblem* zapouzdřuje proměnné

- *unsigned int cities* - celočíselná proměnná udávající počet měst,
- *std::vector<CoordinatesHelper> coordinates* - pole objektů *CoordinatesHelper* velikosti [*cities*] udávající souřadnice měst, kde na *i*-té pozici se nachází souřadnice *i*-tého města.

Třída *TravellingSalesmanProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení symetrické varianty problému slouží metoda *TravellingSalesmanProblem::evaluateSTSP*. Diagram třídy *TravellingSalesmanProblem* je uveden na obrázku (Obr. 8).



Obr. 8. Diagram třídy *TravellingSalesmanProblem*

3.3.7 Třída *CapacitatedVehicleRoutingProblem*

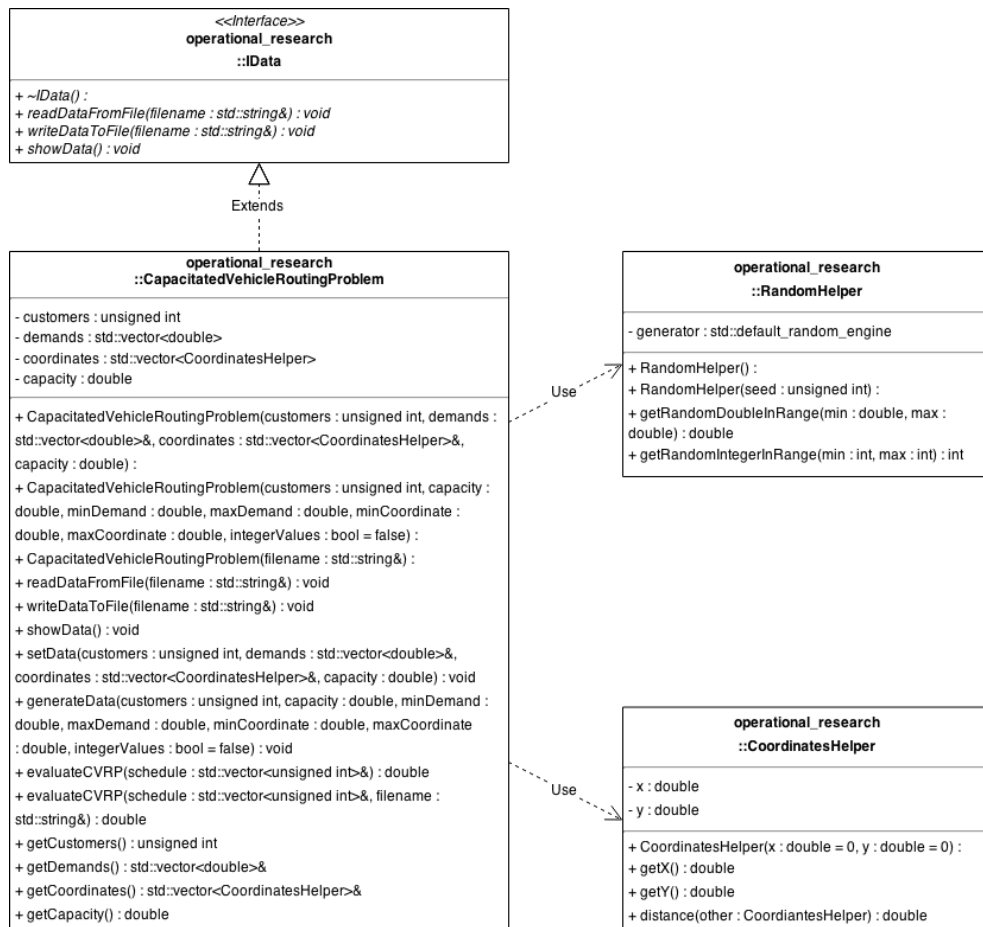
Pro kapacitní rozvozní problém, jehož popis je součástí kapitoly 2.7, byla navržena třída *CapacitatedVehicleRoutingProblem*. Tato třída je odvozena od rozhraní *IData* a využívá pomocné třídy *RandomHelper* a *CoordinatesHelper*.

Třída *CapacitatedVehicleRoutingProblem* zapouzdřuje proměnné

- *unsigned int customers* - celočíselná proměnná udávající počet zákazníků,

- *std::vector<double> demands* - pole reálných hodnot velikosti [*customers*] udávající požadavky zákazníků, kde hodnota na *i*-té pozici odpovídá požadavku *i*-tého zákazníka,
- *std::vector<CoordinatesHelper> coordinates* - pole objektů *CoordinatesHelper* velikosti [*customers* + 1] udávající souřadnice depa a zákazníků, kde na pozici 0 se nachází souřadnice depa, na dalších pozicích *i* = 2, 3, ..., *customers* + 1 se nachází souřadnice (*i* - 1)-tého zákazníka,
- *double capacity* - reálná proměnná udávající kapacitu vozidla.

Třída *CapacitatedVehicleRoutingProblem* dále poskytuje stejné metody jako v případě obecného problému *Problem*, které však přijímají odpovídající parametry. Pro vyhodnocení základní varianty problému slouží metoda *CapacitatedVehicleRoutingProblem::evaluateCVRP*. Diagram třídy *CapacitatedVehicleRoutingProblem* je uveden na obrázku (Obr. 9).



Obr. 9. Diagram třídy *CapacitatedVehicleRoutingProblem*

3.4 Návrh struktury souborů dat problémů

Jedním z požadavků na knihovnu je možnost provádění operace import/export dat pro jednotlivé problémy knihovny. Aby bylo možné splnit tento požadavek, je nutné provést návrh struktury souborů dat pro jednotlivé problémy.

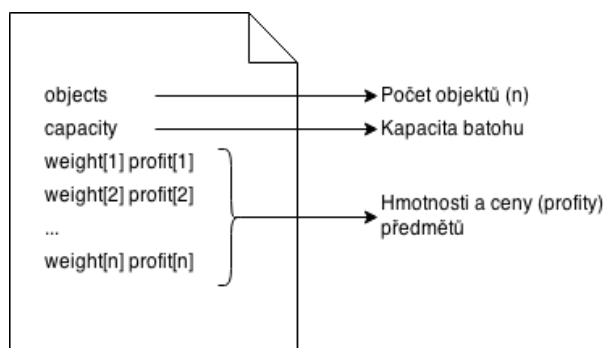
V operačním výzkumu je kvalita algoritmů a heuristik testována jejich aplikací na soubory standardních testovacích (benchmarkových) dat problémů. Testovací data problémů je možné najít na Internetu na různých webových stránkách zabývajících se problémy operačního výzkumu. Nejznámější z těchto webových stránek je OR-Library - kolekce souborů testovacích dat pro různé problémy, kterou je možné najít na adrese <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>.

Problémem souborů testovacích dat je skutečnost, že každý autor si vytváří svoji vlastní strukturu. Není tak možné vytvořit univerzální strukturu, která by byla vhodná pro libovolný soubor testovacích dat problému. Jednou z možností řešení je návrh vlastní struktury souborů testovacích dat. V případě potřeby použití souborů testovacích dat s jinou strukturou je pak nutné tyto soubory upravit tak, aby vyhovovaly navržené struktuře.

Dále budou uvedeny navržené struktury souborů dat pro jednotlivé problémy.

3.4.1 Soubor dat pro problém batohu

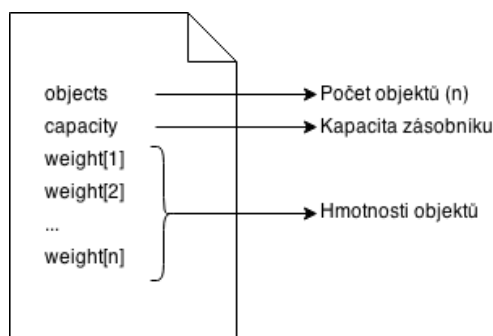
Pro problém batohu byla navržena struktura souboru dat uvedená na obrázku (Obr. 10). Na prvním řádku souboru se nachází počet objektů, na druhém řádku se nachází kapacita batohu. Následuje n řádků pro n objektů, kdy na každém řádku se nachází hmotnost a cena objektu.



Obr. 10. Struktura souboru dat problému batohu

3.4.2 Soubor dat pro problém naplňování zásobníku

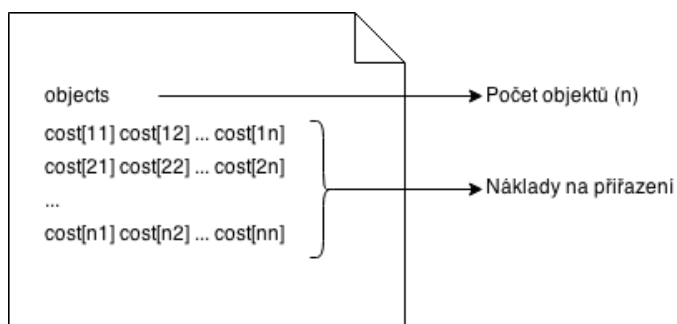
Pro problému naplňování zásobníku byla navržena struktura souboru dat uvedená na obrázku (Obr. 11). Na prvním řádku souboru se nachází počet objektů, na druhém řádku se nachází kapacita zásobníku. Následuje n řádků pro n objektů, kdy na každém řádku se nachází hmotnost objektu.



Obr. 11. Struktura souboru dat pro
blému naplňování zásobníku

3.4.3 Soubor dat pro lineární přiřazovací problém

Pro lineární přiřazovací problém byla navržena struktura souboru dat uvedená na obrázku (Obr. 12). Na prvním řádku souboru se nachází počet objektů. Následuje n řádků pro n objektů, kdy na každém řádku se nachází n nákladů na přiřazení.

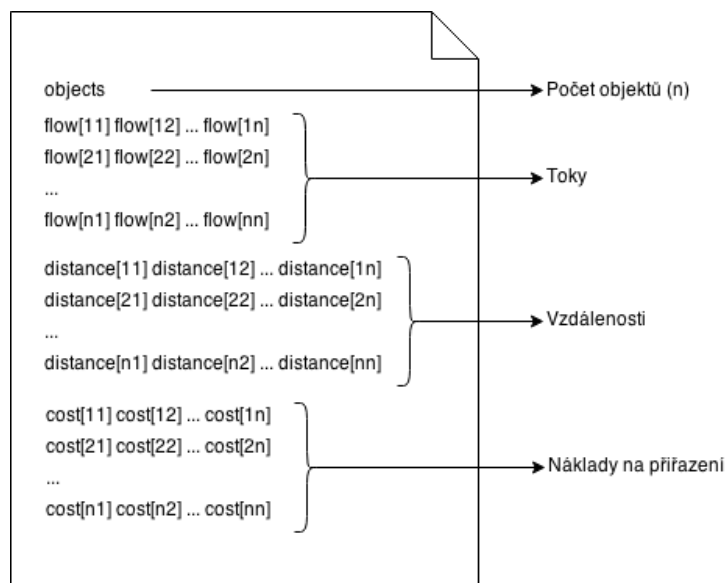


Obr. 12. Struktura souboru dat lineárního přiřazo-
vacího problému

3.4.4 Soubor dat pro kvadratický přiřazovací problém

Pro kvadratický přiřazovací problém byla navržena struktura souboru dat uvedená na obrázku (Obr. 13). Na prvním řádku souboru se nachází počet objektů. Následuje n řádků pro n objektů, kdy na každém řádku se nachází n toků, n řádků pro n objektů, kdy na každém

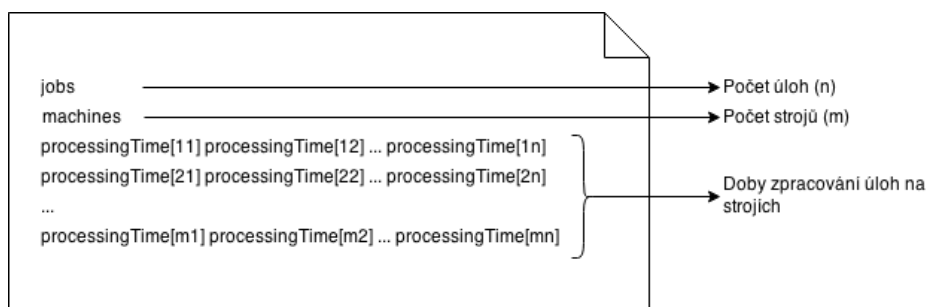
řádku se nachází n vzdáleností, a n řádků pro n objektů, kdy na každém řádku se nachází n nákladů na přiřazení.



Obr. 13. Struktura souboru dat kvadratického přiřazovacího problému

3.4.5 Soubor dat pro problém rozvrhování proudové výroby

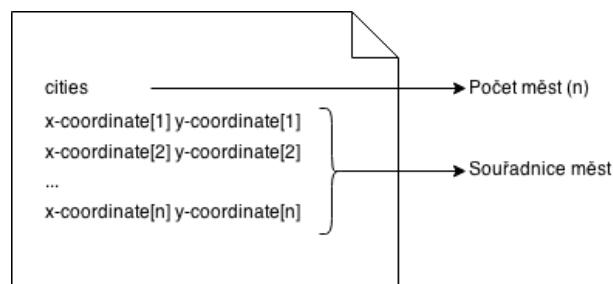
Pro problém rozvrhování proudové výroby byla navržena struktura souboru dat uvedená na obrázku (Obr. 14). Na prvním řádku souboru se nachází počet úloh, na druhém řádku souboru se nachází počet strojů. Následuje m řádků pro m strojů, kdy na každém řádku se nachází n dob zpracování pro n úloh.



Obr. 14. Struktura souboru dat problému rozvrhování proudové výroby

3.4.6 Soubor dat pro problém obchodního cestujícího

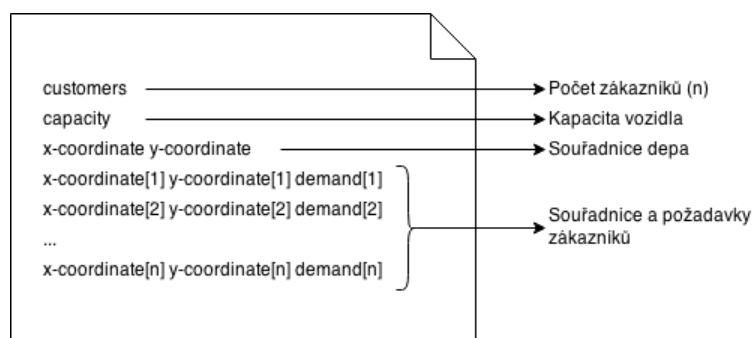
Pro problém obchodního cestujícího byla navržena struktura souboru dat uvedená na obrázku (Obr. 15). Na prvním řádku souboru se nachází počet měst. Následuje n řádků pro n měst, kdy na každém řádku se nachází souřadnice x a y města.



Obr. 15. Struktura souboru dat problému obchodního cestujícího

3.4.7 Soubor dat pro kapacitní rozvozní problém

Pro kapacitní rozvozní problém byla navržena struktura souboru dat uvedená na obrázku (Obr. 16). Na prvním řádku souboru se nachází počet zákazníků, na druhém řádku se nachází kapacita vozidla, na třetím řádku se nachází souřadnice x a y depa. Následuje n řádků pro n zákazníků, kdy na každém řádku se nachází souřadnice x a y zákazníka a požadavek zákazníka.

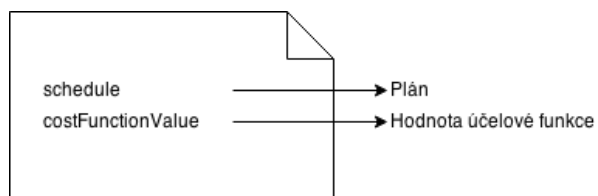


Obr. 16. Struktura souboru dat kapacitního rozvozního problému

3.5 Návrh struktury souboru dat výsledků

Jedním z požadavků na knihovnu je možnost provádění operace export výsledků pro jednotlivé problémy knihovny. Výsledkem se rozumí plán problému a odpovídající hodnota účelové funkce.

Pro export výsledků byla navržena struktura souboru dat uvedená na obrázku (Obr. 17). Na prvním řádku souboru se nachází plán problému, na druhém řádku se nachází odpovídající hodnota účelové funkce.



Obr. 17. Struktura souboru dat výsledků

4 REALIZACE KNIHOVNY OPERAČNÍHO VÝZKUMU

Tato kapitola se zabývá realizací knihovny na základě návrhu tříd, který byl proveden v předchozí kapitole.

Jak již bylo uvedeno, každý problém knihovny má vlastní třídu, která zapouzdřuje proměnné problému a poskytuje metody pro práci s těmito proměnnými. Třída problému je odvozena od rozhraní *IData*, které poskytuje čistě virtuální metody pro čtení/zápis dat ze/do souboru a zobrazení dat problému do konzole. Tyto čistě virtuální metody je nutné implementovat v každé odvozené třídě problému.

Jelikož je princip implementace jednotlivých tříd problémů stejný, byl vybrán jeden z problémů (problém rozvrhování proudové výroby), u kterého je proveden podrobný popis implementace.

4.1 Realizace třídy *FlowShopSchedulingProblem*

Pro problém rozvrhování proudové výroby byla vytvořena třída *FlowShopSchedulingProblem*. Tato třída zapouzdřuje proměnné

- *unsigned int jobs* - celočíselná proměnná udávající počet úloh,
- *unsigned int machines* - celočíselná proměnná udávající počet strojů,
- *std::vector<std::vector<double>> processingTimes* - matice reálných hodnot rozměru [*machines* × *jobs*] udávající doby zpracování úloh na strojích, kde hodnota na *i*-tém řádku a *j*-tém sloupci odpovídá době zpracování *j*-té úlohy na *i*-tém stroji.

4.1.1 Konstruktor *FlowShopSchedulingProblem*

Pro vytvoření instance třídy *FlowShopSchedulingProblem* jsou dostupné 3 varianty konstruktoru

- *FlowShopSchedulingProblem::FlowShopSchedulingProblem(unsigned int jobs, unsigned int machines, const std::vector<std::vector<double>> & processingTimes)*,
- *FlowShopSchedulingProblem::FlowShopSchedulingProblem(unsigned int jobs, unsigned int machines, double minProcessingTime, double maxProcessingTime, bool integerValues)*,

- *FlowShopSchedulingProblem::FlowShopSchedulingProblem(const std::string& filename).*

První varianta konstruktoru slouží pro vytvoření instance třídy *FlowShopSchedulingProblem* ze známých dat. Tento konstruktorem zajišťuje volání metody *FlowShopSchedulingProblem::setData* pro nastavení dat, které jsou jako parametry předávány parametry konstruktoru.

Druhá varianta konstruktoru slouží pro vytvoření instance třídy *FlowShopSchedulingProblem* s generováním dat. Tento konstruktorem zajišťuje volání metody *FlowShopSchedulingProblem::generateData* pro generování dat dle požadavků, které jsou jako parametry předávány parametry konstruktoru.

Třetí varianta konstruktoru slouží pro vytvoření instance třídy *FlowShopSchedulingProblem* s načtením dat ze souboru. Tento konstruktorem zajišťuje volání metody *FlowShopSchedulingProblem::readDataFromFile* pro načtení dat ze souboru, jehož název je předáván jako parametr konstruktoru.

4.1.2 Metoda *setData*

Metoda *void FlowShopSchedulingProblem::setData(unsigned int jobs, unsigned int machines, const std::vector<std::vector<double>>& processingTimes)* slouží pro nastavení dat problému. Tato metoda přijímá parametry

- *unsigned int jobs* - celočíselná proměnná udávající počet úloh,
- *unsigned int machines* - celočíselná proměnná udávající počet strojů,
- *std::vector<std::vector<double>> processingTimes* - matice reálných hodnot rozměru $[machines \times jobs]$ udávající doby zpracování úloh na strojích, kde hodnota na i -tém řádku a j -tém sloupci odpovídá době zpracování j -té úlohy na i -tém stroji.

Před samotným nastavením dat problému se provádí kontrola vstupních parametrů. Kontrola je úspěšná, jestliže počet úloh *jobs* je větší než 0, počet strojů *machines* je větší než 0, matice dob zpracování úloh na strojích *processingTimes* je rozměru $[machines \times jobs]$ a doby zpracování úloh na strojích jsou větší než 0. V případě úspěšné kontroly dojde k nastavení dat problému, v opačném případě je vyvolána výjimka *std::invalid_argument* s odpovídajícím popisem chyby.

4.1.3 Metoda *generateData*

Metoda `void FlowShopSchedulingProblem::generateData(unsigned int jobs, unsigned int machines, double minProcessingTime, double maxProcessingTime, bool integerValues)` slouží pro vygenerování dat problému dle požadavků. Tato metoda přijímá parametry

- *unsigned int jobs* - celočíselná proměnná udávající počet úloh,
- *unsigned int machines* - celočíselná proměnná udávající počet strojů,
- *double minProcessingTime* - reálná hodnota udávající minimální dobu zpracování úlohy na stroji,
- *double maxProcessingTime* - reálná hodnota udávající maximální dobu zpracování úlohy na stroji,
- *bool integerValues* - logická hodnota udávající typ generovaných dat - reálná data (*false*), celočíselná data (*true*).

Před samotným vygenerováním a nastavením dat problému se provede kontrola typu generovaných dat. Jestliže proměnná *integerValues* nabývá hodnoty *true*, meze pro generování dob zpracování úloh na strojích se přetypují z reálné hodnoty na celočíselnou hodnotu. Následuje kontrola vstupních parametrů. Kontrola je úspěšná, jestliže počet úloh *jobs* je větší než 0, počet strojů *machines* je větší než 0, minimální doba zpracování úlohy na stroji *minProcessingTime* a maximální doba zpracování úlohy na stroji *maxProcessingTime* jsou větší než 0. V případě úspěšné kontroly následuje generování dat (doby zpracování úloh na strojích), v opačném případě je vyvolána výjimka `std::invalid_argument` s odpovídajícím popisem chyby. Pro generování se využívá pomocná třída *RandomHelper*, která poskytuje metody pro generování reálných a celočíselných hodnot v daném intervalu. Zda budou generovány reálné nebo celočíselné hodnoty se rozhodne na základě hodnoty proměnné *integerValues*. Po vygenerování dat jsou data nastavena.

4.1.4 Metoda *readDataFromFile*

Metoda `void FlowShopSchedulingProblem::readDataFromFile(const std::string& filename)`, která je implementací virtuální metody *IData::readDataFromFile* rozhraní *IData*, slouží pro načtení dat problému ze souboru. Tato metoda přijímá jediný parametr *filename* - řetězec udávající název souboru, ze kterého má být provedeno čtení dat.

Nejprve je provedeno otevření vstupní streamu souboru s názvem *filename* pomocí metody *std::ifstream::open*. Tato metoda zajistí otevření souboru a jeho asociaci s objektem streamu pro možnost provádění operací nad jeho obsahem. Následně je provedena kontrola, zda je stream asociován se souborem, pomocí metody *std::ifstream::is_open*. V případě chyby je vyvolána výjimka *std::runtime_error* s odpovídajícím popisem chyby. Následuje čtení dat ze souboru, které se řídí návrhem struktury souboru dat problému. Soubor je čten po řádcích pomocí metody *std::getline*, která jako parametry přijímá stream, ze kterého má být provedeno čtení, a řetězec, do kterého má být řádka přečtena. Po načtení řádky je tato řádka umístěna do *std::istringstream* a následně parsována. Prázdné řádky souboru jsou ignorovány. V případě chyby při čtení nebo parsování je vyvolána výjimka *std::runtime_error* s odpovídajícím popisem chyby. Po dokončení čtení a parsování dat jsou data nastavena. Stream souboru je uzavřen automaticky.

4.1.5 Metoda *writeDataToFile*

Metoda *void FlowShopSchedulingProblem::writeDataToFile(const std::string& filename)*, která je implementací virtuální metody *IData::writeDataToFile* rozhraní *IData*, slouží pro zápis dat problému do souboru. Tato metoda přijímá jediný parametr *filename* - řetězec udávající název souboru, do kterého má být proveden zápis dat.

Nejprve je provedeno otevření výstupního streamu souboru s názvem *filename* pomocí metody *std::ofstream::open*. Tato metoda zajistí otevření souboru a jeho asociaci s objektem streamu pro možnost provádění operací nad jeho obsahem. Následně je provedena kontrola, zda je stream asociován se souborem, pomocí metody *std::ofstream::is_open*. V případě chyby je vyvolána výjimka *std::runtime_error* s odpovídajícím popisem chyby. Následuje zápis dat do souboru, který se řídí návrhem struktury souboru dat problému - na první řádek souboru je zapsán počet úloh, na druhý řádek souboru je zapsán počet strojů, na další řádky je zapsána matice dob zpracování úloh na strojích. V případě chyby při zápisu je vyvolána výjimka *std::runtime_error* s odpovídajícím popisem chyby. Stream souboru je uzavřen automaticky.

4.1.6 Metoda *showData*

Metoda *void FlowShopSchedulingProblem::showData()*, která je implementací virtuální metody *IData::showData* rozhraní *IData*, slouží pro zobrazení dat problému do konzole.

Tato metoda zobrazí do konzole počet úloh, počet strojů a matici dob zpracování úloh na strojích.

4.1.7 Metoda *evaluate*

Třída *FlowShopSchedulingProblem* poskytuje metody pro vyhodnocení různých variant problému rozvrhování proudové výroby. Pro permutační problém rozvrhování proudové výroby jsou dostupné metody

- *double FlowShopSchedulingProblem::evaluatePFSSP(const std::vector<unsigned int> & schedule),*
- *double FlowShopSchedulingProblem::evaluatePFSSP(const std::vector<unsigned int> & schedule, const std::string & filename).*

První varianta metody slouží pro stanovení hodnoty účelové funkce permutačního problému rozvrhování proudové výroby pro zvolený plán *schedule*. Před samotným výpočtem hodnoty účelové funkce je provedena kontrola plánu. Kontrola je úspěšná, jestliže zvolený plán je permutací úloh. Tato kontrola je provedena pomocí metoda *std::is_permutation*. V případě úspěšné kontroly dojde ke stanovení hodnoty účelové funkce problému, v opačném případě je vyvolána výjimka *std::invalid_argument* s odpovídajícím popisem chyby. Stanovená hodnota účelové funkce je vrácena jako návratová hodnota metody.

Druhá varianta metody slouží taktéž pro stanovení hodnoty účelové funkce permutačního problému rozvrhování proudové výroby pro zvolený plán *schedule* (je provedeno volání první varianty metody). Kromě stanovení hodnoty účelové funkce je však proveden také zápis výsledku do souboru s názvem *filename*. Výsledkem se rozumí zvolený plán a odpovídající hodnota účelové funkce. Princip zápisu výsledku do souboru je stejný jako v případě zápisu dat problému, proto již nebude dále popisován. Na první řádek souboru je zapsán zvolený plán *schedule*, na druhý řádek pak odpovídající hodnota účelové funkce. V případě chyby spojené s otevíráním souboru nebo zápisem do souboru je vyvolána výjimka *std::runtime_error* s odpovídajícím popisem chyby.

Obdobné metody, ale s jinou účelovou funkcí, jsou dostupné také pro problém rozvrhování proudové výroby s blokováním (*FlowShopSchedulingProblem::evaluateFSSPB*) resp. pro problém rozvrhování proudové výroby s nulovým zpožděním (*FlowShopSchedulingProblem::evaluateFSSPNW*).

4.2 Realizace pomocných tříd

4.2.1 Třída *RandomHelper*

Třída *RandomHelper* je pomocnou třídou pro generování pseudonáhodných hodnot v zadaném intervalu. Tato třída využívá generátor *std::default_random_engine*.

Konstruktor *RandomHelper*

Pro vytvoření instance třídy *RandomHelper* jsou dostupné 2 varianty konstruktoru

- *RandomHelper::RandomHelper()*,
- *RandomHelper::RandomHelper(unsigned int seed)*.

První varianta konstruktoru provede náhodnou inicializaci (seed) generátoru pomocí metody *std::default_random_engine::seed*, která přijímá jako parametr hodnotu seed. K tomu účelu se využívá počet mikrosekund od 1.1.1970 získaných pomocí *std::chrono::system_clock::now().time_since_epoch().count()*.

Druhá varianta konstruktoru provede inicializaci (seed) generátoru na zvolenou hodnotu seed.

Metoda *getRandomDoubleInRange*

Metoda *double RandomHelper::getRandomDoubleInRange(double min, double max)* slouží pro vygenerování reálné hodnoty v intervalu $\langle min; max \rangle$. K tomuto účelu se využívá rozdělení *std::uniform_real_distribution<double>*.

Metoda *getRandomIntegerInRange*

Metoda *int RandomHelper::getRandomIntegerInRange(int min, int max)* slouží pro vygenerování celočíselné hodnoty v intervalu $\langle min; max \rangle$. K tomuto účelu se využívá rozdělení *std::uniform_int_distribution<int>*.

4.2.2 Třída *CoordinatesHelper*

Třída *CoordinatesHelper* je pomocnou třídou pro práci se souřadnicemi.

Konstruktor *CoordinatesHelper*

Pro vytvoření instance třídy *CoordinatesHelper* je dostupná konstruktor *CoordinatesHelper::CoordinatesHelper(double x, double y)*, který přijímá jako parametry hodnoty souřadnic x a y . Jako defaultní hodnota jsou uvažovány souřadnice $[0; 0]$.

Metoda *distance*

Metoda *double CoordinatesHelper::distance(CoordinatesHelper other)* slouží pro výpočet vzdálenosti mezi aktuálními a jinými souřadnicemi, které jsou předávány jako parametr metody. Návrátovou hodnotou metody je euklidovská vzdálenost dvou bodů (souřadnic).

Metoda *getX*

Metoda *double CoordinatesHelper::getX() const* vrací aktuální souřadnici *x*.

Metoda *getY*

Metoda *double CoordinatesHelper::getY() const* vrací aktuální souřadnici *y*.

4.2.3 Třída *ScheduleHelper*

Třída *ScheduleHelper* je pomocnou třídou pro práci (generování, ukládání, načítání) s plánem problémů. Tato třída využívá generátor *std::default_random_engine*.

Konstruktor *ScheduleHelper*

Konstruktor *ScheduleHelper::ScheduleHelper()* provede náhodnou inicializaci (seed) generátoru pomocí metody *std::default_random_engine::seed*, která přijímá jako parametr hodnotu seed. K tomu účelu se využívá počet mikrosekund od 1.1.1970 získaných pomocí *std::chrono::system_clock::now().time_since_epoch().count()*.

Metoda *generatePermutationSchedule*

Metoda *std::vector<unsigned int> ScheduleHelper::generatePermutationSchedule (unsigned int n)* slouží pro vygenerování plánu ve formě permutace množiny $\{1, 2, \dots, n\}$, kde hodnota *n* je předávána jako parametr metody. Před samotným vygenerováním plánu je provedena kontrola. Kontrola je úspěšná, jestliže předávaná hodnota *n* je větší než 0, v opačném případě je vyvolána výjimka *std::invalid_argument* s odpovídajícím popisem chyby. Následně je vytvořeno pole hodnot 1, 2, ..., *n*, jehož prvky jsou přeskládány pomocí metody *std::shuffle*. Pole přeskládaných hodnot (plán) je návratovou hodnotou metody.

Metoda *generateSelectionSchedule*

Metoda *std::vector<unsigned int> ScheduleHelper::generateSelectionSchedule(unsigned int n)* slouží stejně jako metoda *ScheduleHelper::generatePermutationSchedule* pro vygenerování plánu ve formě permutace množiny $\{1, 2, \dots, n\}$, kde hodnota *n* je předávána jako

parametr metody. Z tohoto plánu je ale odstraněno 0 až $n - 1$ položek pomocí metody *erase* volané nad plánem. Takto upravený plán je návratovou hodnotou metody.

Metoda *readScheduleFromFile*

Metoda `std::vector<unsigned int> ScheduleHelper::readScheduleFromFile(const std::string& filename)` slouží pro čtení plánu ze souboru. Tato metoda přijímá jediný parametr *filename* - název souboru, ze kterého má být provedeno čtení. Princip čtení je stejný jako v případě čtení dat problému, proto nebude dále popisován. Plán je čten z prvního řádku souboru. Případné prázdné řádky jsou ignorovány. V případě chyby spojené s otevíráním souboru nebo čtením plánu je vyvolána výjimka `std::runtime_error` s odpovídajícím popisem chyby. V případě úspěšného přečtení plánu je tento plán návratovou hodnotou metody.

Metoda *writeScheduleToFile*

Metoda `void ScheduleHelper::writeScheduleToFile(const std::vector<unsigned int>& schedule, const std::string& filename)` slouží pro zápis plánu do souboru. Tato metoda přijímá parametry *schedule* - plán a *filename* - název souboru, do kterého má být proveden zápis. Před samotným zápisem je provedena kontrola plánu. Kontrola je úspěšná, jestliže plán není prázdný, v opačném případě je vyvolána výjimka `std::invalid_argument` s odpovídajícím popisem chyby. Princip zápisu je stejný jako v případě zápisu dat problému, proto nebude dále popisován. Plán je zapsán na první řádek souboru. V případě chyby spojené s otevíráním souboru nebo zápisem plánu je vyvolána výjimka `std::runtime_error` s odpovídajícím popisem chyby.

5 TESTOVÁNÍ KNIHOVNY OPERAČNÍHO VÝZKUMU NA SADĚ BENCHMARKOVÝCH DAT

Tato kapitola se zabývá testováním realizované knihovny na sadě testovacích (benchmarkových) dat.

Jak již bylo uvedeno, v operačním výzkumu je kvalita algoritmů a heuristik testována jejich aplikací na soubory standardních testovacích (benchmarkových) dat. Testovací data je možné najít na Internetu na různých webových stránkách zabývajících se problémy operačního výzkumu. Problémem těchto testovacích dat je skutečnost, že každý autor si vytváří svoji vlastní strukturu. Z tohoto důvodu byla navržena a realizována vlastní struktura souborů dat. Soubory testovacích dat jiných autorů je, v případě odlišné struktury, potřeba upravit tak, aby vyhovovaly struktuře podporované knihovnou.

Veškeré soubory testovacích dat použité pro testování problémů implementovaných v knihovně jsou dostupné na přiloženém CD.

5.1 Problém batohu

Pro testování problému batohu byly použity soubory testovacích dat uvedené v tabulce (Tab. 1). Jedná se o vlastní soubory testovacích dat, které byly vygenerovány knihovnou, jelikož pro základní verzi problému batohu nebyly nalezeny soubory testovacích dat. Bez znalosti plánu a odpovídající hodnoty účelové funkce není možné provedení ověření výstupu knihovny. Z tohoto důvodu bylo pro soubory testovacích dat provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulce (Tab. 1).

Tab. 1. Výsledky testování problému batohu

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
kp25	kp25_schedule	797
kp50	kp50_schedule	2736
kp75	kp75_schedule	3288
kp100	kp100_schedule	5509
kp200	kp200_schedule	9318

5.2 Problém naplňování zásobníku

Pro testování problému naplňování zásobníku byly použity soubory testovacích dat uvedené v tabulce (Tab. 2). Jedná se o soubory testovacích dat, které jsou dostupné na adrese <http://www.wiwi.uni-jena.de/Entscheidung/binpp/bin1dat.htm>. U těchto souborů testovacích dat však není uvedeno optimální řešení resp. jakékoliv řešení a odpovídající hodnota účelové funkce, aby mohlo být provedeno srovnání s výstupem knihovny. Z tohoto důvodu bylo pro soubory testovacích dat provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulce (Tab. 2).

Tab. 2. Výsledky testování problému naplňování zásobníku

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
N1C1W1_A	N1C1W1_A_schedule	30
N1C1W1_B	N1C1W1_B_schedule	35
N1C1W1_C	N1C1W1_C_schedule	26
N1C1W1_D	N1C1W1_D_schedule	32
N1C1W1_E	N1C1W1_E_schedule	31

5.3 Lineární přiřazovací problém

Pro testování lineárního součtového přiřazovacího problému byly použity soubory testovacích dat uvedené v tabulce (Tab. 3). Jedná se o vlastní soubory testovacích dat, které byly vygenerovány knihovnou operačního výzkumu, jelikož pro lineární součtový přiřazovací problém nebyly nalezeny soubory testovacích dat. Bez znalosti plánu a odpovídající hodnoty účelové funkce není možné provedení ověření výstupu knihovny operačního výzkumu. Z tohoto důvodu bylo pro soubory testovacích dat provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulce (Tab. 3).

Tab. 3. Výsledky testování lineárního součtového přiřazovacího problému

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
lap10	lap10_schedule	237
lap15	lap15_schedule	421
lap20	lap20_schedule	1031
lap30	lap30_schedule	1312
lap50	lap50_schedule	2581

Výše uvedené soubory testovacích dat byly použity také pro testování lineárního úzkoprofilového přiřazovacího problému. V tomto případě bylo opět provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulce (Tab. 4).

Tab. 4. Výsledky testování lineárního úzkoprofilového přiřazovacího problému

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
lap10	lap10_schedule	42
lap15	lap15_schedule	48
lap20	lap20_schedule	100
lap30	lap30_schedule	99
lap50	lap50_schedule	95

5.4 Kvadratický přiřazovací problém

Pro testování kvadratického přiřazovacího problému byly použity soubory testovacích dat uvedené v tabulce (Tab. 5). Jedná se o soubory testovacích dat, které jsou dostupné na adrese <http://anjos.mgi.polymtl.ca/qaplib/inst.html> včetně jejich optimálního řešení a odpovídající hodnoty účelové funkce. Tyto testovací datové soubory byly vyhodnoceny knihovnou operačního výzkumu pro daný plán, kdy byla získána hodnota účelové funkce pro da-

ný plán. Takto získaná hodnota účelové funkce byla srovnána s hodnotou účelové funkce uvedenou u testovacích dat. Získané výsledky jsou uvedeny v tabulce (Tab. 5).

Tab. 5. Výsledky testování kvadratického přiřazovacího problému

Testovací data	Plán	Hodnota účelové funkce	
		stanovená knihovnou	uvedená u testovacích dat
had12	had12_schedule	1652	1652
had14	had14_schedule	2724	2724
had16	had16_schedule	3720	3720
had18	had18_schedule	5358	5358
had20	had20_schedule	6922	6922

Z výsledků uvedených v tabulce (Tab. 5) je zřejmé, že ve všech případech je hodnota účelové funkce stanovená knihovnou shodná s hodnotou účelové funkce uvedené u testovacích dat.

Výše uvedené soubory testovacích dat byly použity také pro testování kvadratického úzkoprofilového přiřazovacího problému. V tomto případě bylo provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulce (Tab. 6).

Tab. 6. Výsledky testování kvadratického úzkoprofilového přiřazovacího problému

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
had12	had12_schedule	27
had14	had14_schedule	42
had16	had16_schedule	49
had18	had18_schedule	63
had20	had20_schedule	70

5.5 Problém rozvrhování proudové výroby

Pro testování permutačního problému rozvrhování proudové výroby byly použity soubory testovacích dat uvedené v tabulce (Tab. 7). Jedná se o soubory testovacích dat, které jsou

dostupné na adrese <http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/flowshop1.txt>. U těchto souborů testovacích dat však není uvedeno optimální řešení resp. jakékoliv řešení a odpovídající hodnota účelové funkce aby mohlo být provedeno srovnání s výstupem knihovny operačního výzkumu. Z tohoto důvodu bylo pro soubory testovacích dat provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulce (Tab. 7).

Tab. 7. Výsledky testování permutačního problému rozvrhování proudové výroby

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
car1	car1_schedule	8483
car2	car2_schedule	8042
car3	car3_schedule	8828
car4	car4_schedule	10985
car5	car5_schedule	9532

Výše uvedené soubory testovacích dat byly použity také pro testování problému rozvrhování proudové výroby s blokováním a problému rozvrhování proudové výroby s nulovým zpožděním. V těchto případech bylo opět provedeno pouze stanovení hodnoty účelové funkce pro zvolený plán bez dalšího ověření. Získané výsledky jsou uvedeny v tabulkách (Tab. 8) a (Tab. 9).

Tab. 8. Výsledky testování problému rozvrhování proudové výroby s blokováním

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
car1	car1_schedule	9838
car2	car2_schedule	10104
car3	car3_schedule	10272
car4	car4_schedule	11388

car5	car5_schedule	10814
------	---------------	--------------

Tab. 9. Výsledky testování problému rozvrhování proudové výroby s nulovým zpožděním

Testovací data	Plán	Hodnota účelové funkce stanovená knihovnou
car1	car1_schedule	10907
car2	car2_schedule	11220
car3	car3_schedule	12215
car4	car4_schedule	11726
car5	car5_schedule	12784

5.6 Problém obchodního cestujícího

Pro testování symetrického problému obchodního cestujícího byly použity soubory testovacích dat uvedené v tabulce (Tab. 10). Jedná se o soubory testovacích dat, které jsou dostupné na adrese <http://elib.zib.de/pub/mp-testdata/tsp/tsplib/tsp/index.html>, jejich optimální řešení jsou dostupné na adrese <http://elib.zib.de/pub/mp-testdata/tsp/tsplib/stsp-sol.html>. Tyto testovací datové soubory byly vyhodnoceny knihovnou operačního výzkumu pro daný plán, kdy byla získána hodnota účelové funkce pro daný plán. Takto získaná hodnota účelové funkce byla srovnána s hodnotou účelové funkce uvedenou u testovacích dat. Získané výsledky jsou uvedeny v tabulce (Tab. 10).

Tab. 10. Výsledky testování symetrického problému obchodního cestujícího

Testovací data	Plán	Hodnota účelové funkce	
		stanovená knihovnou	uvedená u testovacích dat
st70	st70_schedule	678,597 (675)	675
pr76	pr76_schedule	108159,438 (108159)	108159
eil51	eil51_schedule	429,983 (426)	426
ch130	ch130_schedule	6110,861 (6110)	6110
berlin52	berlin52_schedule	7544,366 (7542)	7542

V tabulce (Tab. 10) se v případě stanovení hodnoty účelové funkce knihovnou vyskytují 2 hodnoty. První hodnota je standardní výstup knihovny. Je zřejmé, že tato hodnota se liší od hodnoty účelové funkce uvedené u testovacích dat. Tato odlišnost je způsobena zaokrouhlením, jak je uvedeno v popisu testovacích souborů na výše uvedené stránce, které je aplikováno na každý výpočet vzdáleností. Pokud by toto zaokrouhlení bylo zavedeno také v knihovně, pak by výstupem byla hodnota uvedená v závorce, která je shodná s hodnotou účelové funkce uvedené u testovacích dat.

5.7 Kapacitní rozvozní problém

Pro testování kapacitního rozvozního problému byly použity soubory testovacích dat uvedené v tabulce (Tab. 11). Jedná se o soubory testovacích dat, které jsou dostupné na adrese <http://www.coin-or.org/SYMPHONY/branchandcut/VRP/data/index.htm> včetně jejich optimálního řešení a odpovídající hodnoty účelové funkce. Tyto testovací datové soubory byly vyhodnoceny knihovnou operačního výzkumu pro daný plán, kdy byla získána hodnota účelové funkce pro daný plán. Takto získaná hodnota účelové funkce byla srovnána s hodnotou účelové funkce uvedenou u testovacích dat. Získané výsledky jsou uvedeny v tabulce (Tab. 11).

Tab. 11. Výsledky testování kapacitního rozvozního problému

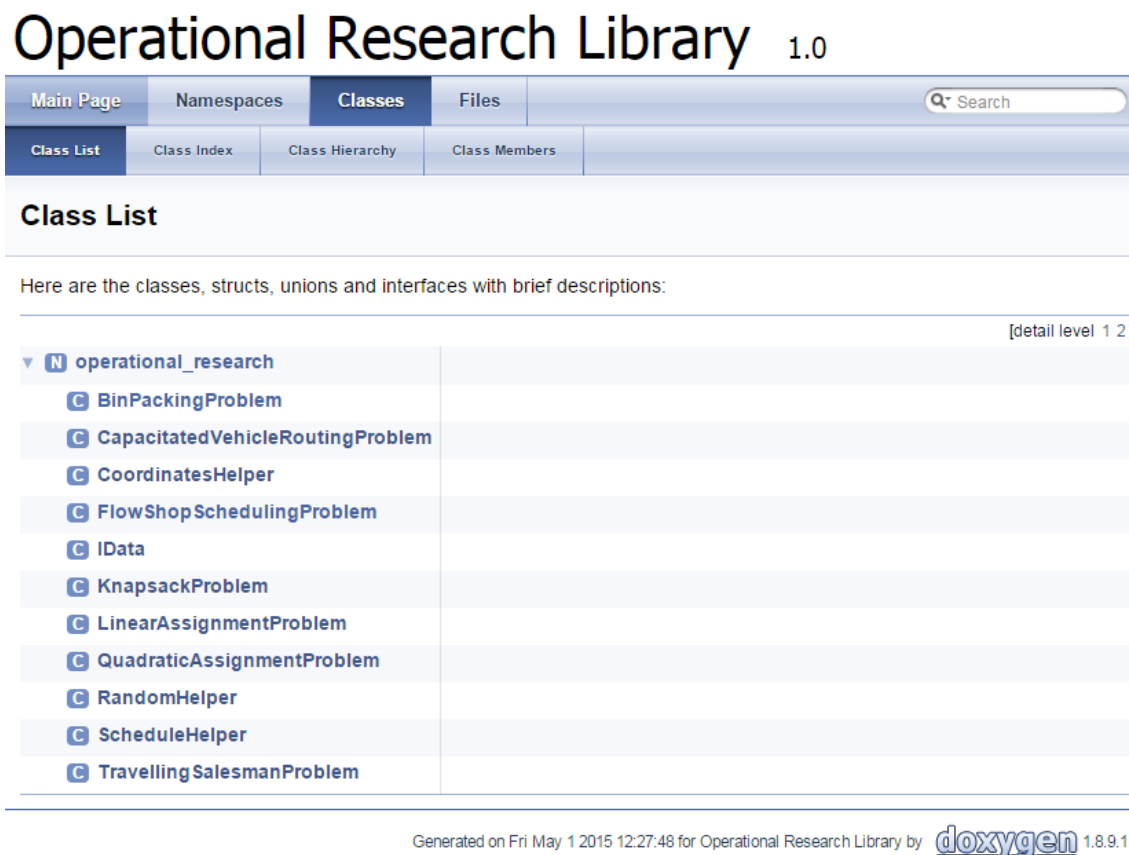
Testovací data	Plán	Hodnota účelové funkce	
		stanovená knihovnou	uvedená u testovacích dat
P-n16-k8	P-n16-k8_schedule	451,947 (450)	450
P-n19-k2	P-n19-k2_schedule	212,657 (212)	212
P-n20-k2	P-n20-k2_schedule	217,416 (216)	216
P-n21-k2	P-n21-k2_schedule	212,712 (211)	211
P-n22-k2	P-n22-k2_schedule	217,852 (216)	216

V tabulce (Tab. 11) se v případě stanovení hodnoty účelové funkce knihovnou vyskytují 2 hodnoty. První hodnota je standardní výstup knihovny. Je zřejmé, že tato hodnota se liší od hodnoty účelové funkce uvedené u testovacích dat. Tato odlišnost je způsobena zaokrouhlením, jak je uvedeno v popisu testovacích souborů na výše uvedené stránce, které je aplikováno na každý výpočet vzdáleností. Pokud by toto zaokrouhlení bylo zavedeno také v knihovně, pak by výstupem byla hodnota uvedená v závorce, která je shodná s hodnotou účelové funkce uvedené u testovacích dat.

6 DOKUMENTACE KNIHOVNY OPERAČNÍHO VÝZKUMU

6.1 Programová dokumentace

Pro tvorbu programové dokumentace knihovny byl využit Doxygen - nástroj pro generování dokumentace z okomentovaných zdrojových kódů. Dokumentace je generovaná jako online dokumentace v HTML. Tato dokumentace je dostupná na přiloženém CD.



Operational Research Library 1.0

Main Page Namespaces **Classes** Files Search

Class List Class Index Class Hierarchy Class Members

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

[detail level 1 2]

▼ N operational_research	
C BinPackingProblem	
C CapacitatedVehicleRoutingProblem	
C CoordinatesHelper	
C FlowShopSchedulingProblem	
C IData	
C KnapsackProblem	
C LinearAssignmentProblem	
C QuadraticAssignmentProblem	
C RandomHelper	
C ScheduleHelper	
C TravellingSalesmanProblem	

Generated on Fri May 1 2015 12:27:48 for Operational Research Library by **doxygen** 1.8.9.1

Obr. 18. Dokumentace knihovny operačního výzkumu - přehled tříd knihovny

6.2 Manuál pro použití knihovny

Kromě programové online dokumentace v HTML byl vytvořen také manuál pro použití knihovny, který je dostupný na přiloženém CD. Tento manuál popisuje způsob použití jednotlivých problémů implementovaných v knihovně. Kromě stručné charakteristiky jednotlivých metod tříd problémů je u každého problému uvedena také funkční ukázka.

7 MOŽNOSTI BUDOUCÍHO ROZŠÍŘENÍ A VYLEPŠENÍ KNIHOVNY OPERAČNÍHO VÝZKUMU

Realizovaná knihovna implementuje vybrané problémy, kterými jsou

- problém batohu,
- problém naplňování zásobníku,
- lineární přiřazovací problém,
- kvadratický přiřazovací problém,
- problém rozvrhování proudové výroby,
- problém obchodního cestujícího,
- kapacitní rozvozní problém.

Knihovnu je v budoucnosti možné rozšířit o nové problémy, stejně tak rozšířit již existující problémy o nové varianty, jelikož knihovna se zaměřuje především na základní varianty problémů. Rozšíření knihovny o nový problém vychází z požadavku na modularitu knihovny.

Rozšíření knihovny o nový problém spočívá ve vytvoření nové třídy odvozené od rozhraní *IData*, implementace čistě virtuálních metod rozhraní, zapouzdření proměnných problému a implementace metod pro práci (nastavení, generování, vyhodnocení) s těmito proměnnými. Je taktéž nutné navrhnout strukturu souboru dat problému, ze které vychází metody pro import/export dat problému ze/do souboru.

V případě rozšíření knihovny o novou variantu již implementovaného problému je možné doplnit novou metodu pro stanovení hodnoty účelové funkce pro tuto variantu problému. Pokud by ale aktuální struktura třídy problému nevyhovovala, je nutné vytvořit třídu novou viz rozšíření o nový problém.

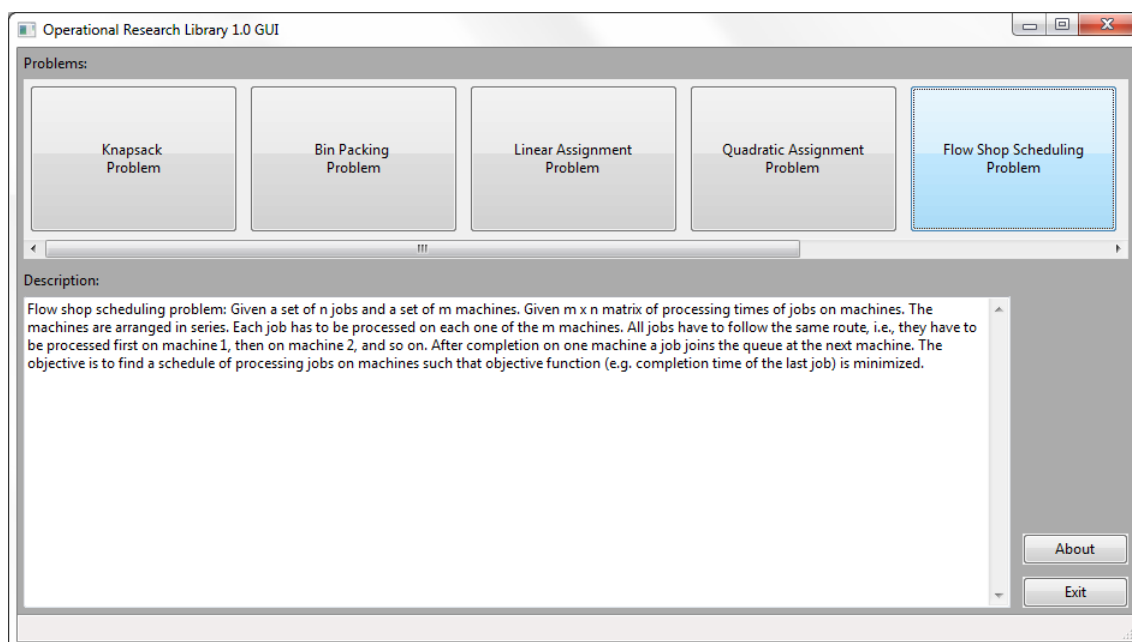
Knihovna podporuje pouze jeden formát souboru dat pro každý problém. V budoucnosti je tak možné rozšířit počet podporovaných formátů, případně realizovat parsery souborů testovacích dat jiných autorů. U každého problému knihovny by pak bylo možné volit mezi několika implementovanými formáty souborů dat, což by usnadnilo použití knihovny.

8 APLIKACE S GRAFICKÝM UŽIVATELSKÝM ROZHRAŇÍM VYUŽÍVAJÍCÍ KNIHOVNY OPERAČNÍHO VÝZKUMU

Z důvodu prezentace výsledků práce, ale také jako ukázka možného využití realizované knihovny, byla vytvořena jednoduchá aplikace s grafickým uživatelským rozhraním (GUI) využívající realizované knihovny. Aplikace s GUI není součástí zadání práce, proto nebude pojednáno o způsobu realizace. K této aplikaci byl vytvořen manuál, který popisuje jednotlivé prvky a způsob ovládání GUI.

Pro realizaci GUI bylo využito

- wxWidgets
 - multiplatformní GUI knihovna pro C++
 - dostupnost: <http://www.wxwidgets.org/>
- wxFormBuilder
 - open source, multiplatformní RAD (rapid application development) nástroj pro wxWidgets
 - dostupnost: <http://sourceforge.net/projects/wxformbuilder/>



Obr. 19. Aplikace s GUI využívající knihovny operačního výzkumu

Aplikace s GUI využívající realizované knihovny, zdrojové soubory a manuál jsou dostupné na příloženém CD.

ZÁVĚR

Cílem diplomové práce bylo ve zvoleném programovém prostředí (C/C++/Mathematica) realizovat komplexní knihovnu nejznámějších a nejstudovanějších problematik operačního výzkumu. Tato knihovna měla být zpracována ve formě účelových funkcí tak, aby mohly být tyto funkce dále v budoucnosti podrobeny testům pomocí různých algoritmů.

V teoretické části práce byla provedena literární rešerše na téma operační výzkum a matematický popis vybraných problémů operačního výzkumu - problém batohu, problém naplňování zásobníku, lineární přiřazovací problém, kvadratický přiřazovací problém, problém rozvrhování proudové výroby, problém obchodního cestujícího, kapacitní rozvozní problém - včetně popisu jejich variant.

Praktická část se zabývá návrhem a realizací knihovny operačního výzkumu. Jako programové prostředí pro realizaci knihovny byl zvolen objektově orientovaný programovací jazyk C++.

Návrh knihovny vychází z požadavků kladených na knihovnu, kdy základním požadavkem je modularita, tj. možnost snadného rozšíření o nový problém resp. variantu již implementovaného problému. Jednotlivé navržené části knihovny jsou řádně popsány a doplněny o UML diagramy.

Na základě návrhu knihovny byla následně provedena programová realizace. Realizace jednotlivých částí knihovny je řádně popsána, v případě problémů knihovny je proveden detailní popis pouze u jednoho vybraného problému, jelikož u ostatních problémů je postup obdobný.

Realizovaná knihovna byla otestována na sadě standardních testovacích (benchmarkových) dat, kdy hodnota účelové funkce stanovená knihovnou byla srovnána se známou hodnotou účelové funkce uvedené u testovacích dat pro daný plán. Pro některé problémy se však nepodařilo najít testovací data, pro jiné problémy se podařilo najít testovací data pouze bez uvedeného plánu a odpovídající hodnoty účelové funkce. V těchto případech byla vytvořena vlastní testovací data resp. využita data bez uvedeného plánu a odpovídající hodnoty účelové funkce a bylo provedeno pouze stanovení hodnoty účelové funkce knihovnou pro zvolený plán bez dalšího srovnání.

K realizované knihovně byla vytvořena programová dokumentace a manuál. Programová dokumentace ve formě online dokumentace v HTML byla vygenerována pomocí nástroje Doxygen z okomentovaných zdrojových kódů knihovny.

V závěru praktické části je pojednáno o možnostech budoucího rozšíření a vylepšení realizované knihovny.

Z důvodu prezentace výsledků práce, ale také jako ukázka možného využití realizované knihovny, byla vytvořena jednoduchá aplikace s GUI využívající realizované knihovny.

Veškeré výstupy práce jsou dostupné na přiloženém CD.

SEZNAM POUŽITÉ LITERATURY

- [1] ÚVOD DO OPTIMÁLNÍHO ROZHODOVÁNÍ. In: *Distance Learning Module for Management Science* [online]. 2007 [cit. 2015-04-21]. Dostupné z: <http://orms.pef.czu.cz/text/cesky/kapitola1.html>
- [2] PROCHÁZKA, Miroslav. *Možnosti využití metod operačního výzkumu ve zdravotnické službě AČR* [online]. Brno, 2007 [cit. 2015-04-21]. Dostupné z: is.muni.cz/th/55291/lf_d/prochazka_disertace.pdf. Disertační práce. Masarykova univerzita, Lékařská fakulta. Vedoucí práce MUDr. Vojtěch Humlíček, Ph.D.
- [3] HILLIER, Frederick S. a Gerald J. LIEBERMAN. *Introduction to operations research*. 9th ed. New York: McGraw-Hill Higher Education, c2010, xxiv, 1047 s. ISBN 978-0-07-337629-5.
- [4] MURTHY, P. *Operations research*. 2nd ed. New Delhi: New Age International, 2007, 705 p. ISBN 978-81-224-2944-2.
- [5] JABLONSKÝ, Josef. *Operační výzkum: kvantitativní metody pro ekonomické rozhodování*. 3. vyd. Praha: Professional Publishing, 2007, 323 s. ISBN 978-80-86946-44-3.
- [6] What is Operations Research?. In: *INFORMS* [online]. [cit. 2015-04-21]. Dostupné z: <https://www.informs.org/About-INFORMS/What-is-Operations-Research>
- [7] ZIMOLA, Bedřich. *Operační výzkum*. 5. vyd. Zlín: Univerzita Tomáše Bati ve Zlíně, 2009, 168 s. ISBN 978-807-3188-788.
- [8] FÁBRY, Jan. *Matematické modelování*. Praha: Professional Publishing, 2011, 180 s. ISBN 978-80-7431-066-9.
- [9] DEMEL, Jiří. *Operační výzkum*. In: *Předměty Katedry řídicí techniky FEL* [online]. 7. března 2011, 148 s., [cit. 2015-02-15]. Dostupné z: https://moodle.dce.fel.cvut.cz/pluginfile.php/5375/mod_resource/content/4/OperacniVyzkum.pdf
- [10] TAHA, Hamdy A. *Operations research: an introduction*. 8th ed. Upper Saddle River, N.J.: Pearson/Prentice Hall, c2007, xx, 813 s. ISBN 01-318-8923-0.
- [11] VOLNÁ, Eva. *Základy softcomputingu* [online]. Ostrava, 2012 [cit. 2015-05-02]. Dostupné z: http://www1.osu.cz/~volna/Zaklady_softcomputingu_skripta.pdf

- [12] KELLERER, Hans, Ulrich PFERSCHY a David PISINGER. *Knapsack problems*. Berlin: Springer, c2010, xx, 546 s. ISBN 978-364-2073-113.
- [13] MARTELLO, Silvano a Paolo TOTH. *Knapsack problems: algorithms and computer implementations*. New York: J. Wiley, c1990, xii, 296 p. ISBN 04-719-2420-2.
- [14] BURKARD, Rainer, Mauro DELL'AMICO a Silvano MARTELLO. *Assignment problems*. Philadelphia: Society for Industrial and Applied Mathematics, c2009, xx, 382 s. ISBN 978-0-89871-663-4.
- [15] PINEDO, Michael L. *Scheduling: theory, algorithms, and systems*. 3rd ed. New York: Springer, c2008, xvii, 671 s. ISBN 978-0-387-78934-7.
- [16] DAVENDRA, Donald. *Chaotic Attributes and Permutative Optimization* [online]. Zlín, 2009 [cit. 2015-04-21]. Dostupné po přihlášení z: http://portal.utb.cz/wps/PA_StagPortletsJSR168/KvalifPraceDownloadServlet?typ=1&adipidno=13543. Disertační práce. Univerzita Tomáše Bati ve Zlíně, Fakulta aplikované informatiky. Vedoucí práce prof. Ing. Zelinka Ivan, Ph.D.
- [17] GUTIN, Gregory. *The traveling salesman problem and its variations*. Dordrecht: Kluwer Academic Publishers, 2002, 830 s. ISBN 14-020-0664-0.
- [18] *Traveling salesman problem, theory and applications: Edited by Donald Davendra* [online]. 2010 [cit. 2015-04-21]. ISBN 978-953-307-426-9. Dostupné z: <http://www.slideshare.net/sachinkheveria/traveling-salesman-problemtheoryandapplications>
- [19] DAVENDRA, Donald. Travelling Salesman Problem. In: *Media Research Lab* [online]. 2014 [cit. 2015-04-21]. Dostupné z: http://mrl.cs.vsb.cz/people/davendra/OVData/Lab_Lectures/Lab%202.pdf
- [20] TOTH, Paolo a Daniele VIGO. *The vehicle routing problem*. Philadelphia: Society for Industrial and Applied Mathematics, c2002, xviii, 367 p. ISBN 08-987-1498-2.
- [21] A GRASP Algorithm Based on New Randomized Heuristic for Vehicle Routing Problem. In: *CIT. Journal of Computing and Information Technology* [online]. 2013, Vol. 21, No. 1 [cit. 2015-04-21]. ISSN 1330-1136. DOI: 10.2498/cit.1002085. Dostupné z: <http://hrcak.srce.hr/file/151987>

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

Seznam symbolů

Význam použitých symbolů je uveden přímo u jejich výskytu v textu.

Seznam zkratk

ATSP	Asymmetric travelling salesman problem
BAP	Balanced assignment problem
BKP	Bounded knapsack problem
BPP	Bin packing problem
CVRP	Capacitated vehicle routing problem
DCVRP	Distance-constrained capacitated vehicle routing problem
<i>d</i> -KP	<i>d</i> -dimensional knapsack problem
DVRP	Distance-constrained vehicle routing problem
FIFO	First in first out
FSSP	Flow shop scheduling problem
FSSPLIS	Flow shop scheduling problem with limited intermediate storage
FSSPNW	Flow shop scheduling problem with no-wait
FSSPUIS	Flow shop scheduling problem with unlimited intermediate storage
GUI	Graphical user interface
KP	Knapsack problem
LAP	Linear assignment problem
LBAP	Linear bottleneck assignment problem
LSAP	Linear sum assignment problem
MCKP	Multiple-choice knapsack problem
MKP	Multiple knapsack problem

MTSP	Multiple travelling salesman problem
QAP	Quadratic assignment problem
QBAP	Quadratic bottleneck assignment problem
RAD	Rapid application development
SC	Soft computing
semi-QAP	Quadratic semi-assignment problem
SSP	Subset sum problem
STSP	Symmetric travelling salesman problem
Sum- k AP	Sum- k assignment problem
TSP	Travelling salesman problem
UKP	Unbounded knapsack problem
UML	Unified modeling language
VRP	Vehicle routing problem
VRPB	Vehicle routing problem with backhauls
VRPPD	Vehicle routing problem with pickup and delivery
VRPTW	Vehicle routing problem with time windows

SEZNAM OBRÁZKŮ

<i>Obr. 1. Rozdělení rozhodovacího procesu [2, s. 24]</i>	12
<i>Obr. 2. Fáze při aplikaci operačního výzkumu [5, s. 11]</i>	18
<i>Obr. 3. Diagram třídy KnapsackProblem</i>	52
<i>Obr. 4. Diagram třídy BinPackingProblem</i>	53
<i>Obr. 5. Diagram třídy LinearAssignmentProblem</i>	54
<i>Obr. 6. Diagram třídy QuadraticAssignmentProblem</i>	56
<i>Obr. 7. Diagram třídy FlowShopSchedulingProblem</i>	57
<i>Obr. 8. Diagram třídy TravellingSalesmanProblem</i>	58
<i>Obr. 9. Diagram třídy CapacitatedVehicleRoutingProblem</i>	59
<i>Obr. 10. Struktura souboru dat problému batohu</i>	60
<i>Obr. 11. Struktura souboru dat problému naplňování zásobníku</i>	61
<i>Obr. 12. Struktura souboru dat lineárního přiřazovacího problému</i>	61
<i>Obr. 13. Struktura souboru dat kvadratického přiřazovacího problému</i>	62
<i>Obr. 14. Struktura souboru dat problému rozvrhování proudové výroby</i>	62
<i>Obr. 15. Struktura souboru dat problému obchodního cestujícího</i>	63
<i>Obr. 16. Struktura souboru dat kapacitního rozvozního problému</i>	63
<i>Obr. 17. Struktura souboru dat výsledků</i>	64
<i>Obr. 18. Dokumentace knihovny operačního výzkumu - přehled tříd knihovny</i>	80
<i>Obr. 19. Aplikace s GUI využívající knihovny operačního výzkumu</i>	82

SEZNAM TABULEK

<i>Tab. 1. Výsledky testování problému batohu</i>	<i>73</i>
<i>Tab. 2. Výsledky testování problému naplňování zásobníku</i>	<i>74</i>
<i>Tab. 3. Výsledky testování lineárního součtového přiřazovacího problému</i>	<i>75</i>
<i>Tab. 4. Výsledky testování lineárního úzkoprofilového přiřazovacího problému</i>	<i>75</i>
<i>Tab. 5. Výsledky testování kvadratického přiřazovacího problému</i>	<i>76</i>
<i>Tab. 6. Výsledky testování kvadratického úzkoprofilového přiřazovacího problému</i>	<i>76</i>
<i>Tab. 7. Výsledky testování permutačního problému rozvrhování proudové výroby.....</i>	<i>77</i>
<i>Tab. 8. Výsledky testování problému rozvrhování proudové výroby s blokováním.....</i>	<i>77</i>
<i>Tab. 9. Výsledky testování problému rozvrhování proudové výroby s nulovým zpožděním</i>	<i>78</i>
<i>Tab. 10. Výsledky testování symetrického problému obchodního cestujícího</i>	<i>78</i>
<i>Tab. 11. Výsledky testování kapacitního rozvozního problému.....</i>	<i>79</i>

SEZNAM PŘÍLOH

P I Obsah přiloženého CD

PŘÍLOHA P I: OBSAH PŘÍLOŽENÉHO CD

Na přiloženém CD se nachází soubor *fulltext.pdf* (tato práce ve formátu PDF) a adresář *prilohy*, jehož struktura je následující:

/prilohy/zadani/

Adresář obsahující naskenované zadání diplomové práce.

/prilohy/knihovna_operacniho_vyzkumu/aplikace/

Adresář obsahující aplikaci s GUI využívající realizované knihovny.

/prilohy/knihovna_operacniho_vyzkumu/dokumentace/

Adresář obsahující programovou dokumentaci knihovny, manuál knihovny a manuál aplikace s GUI.

/prilohy/knihovna_operacniho_vyzkumu/testovaci_data/

Adresář obsahující testovací (benchmarková) data použita při testování knihovny.

/prilohy/knihovna_operacniho_vyzkumu/zdrojove_soubory_gui/

Adresář obsahující zdrojové soubory aplikace s GUI.

/prilohy/knihovna_operacniho_vyzkumu/zdrojove_soubory_knihovny/

Adresář obsahující zdrojové soubory knihovny.