

Vysoká škola ekonomická v Praze
Fakulta managementu v Jindřichově Hradci

Diplomová práce

Bc. Blanka Bártová
2016



Vysoká škola ekonomická v Praze

Fakulta managementu v Jindřichově Hradci

Katedra exaktních metod

Optimalizace výroby s využitím pokročilých metod pro podporu rozhodování

Autor diplomové práce:	Bc. Blanka Bártová
Vedoucí diplomové práce:	doc. Ing. Vladislav Bína, Ph.D.
Rok obhajoby:	2016

Čestné prohlášení

*Prohlašuji, že diplomovou práci na téma
„Optimalizace výroby s využitím pokročilých metod pro podporu rozhodování“
jsem vypracovala samostatně a veškerou použitou literaturu a další prameny
jsem řádně označila a uvedla v přiloženém seznamu.*

V Jindřichově Hradci dne 3. 5. 2016

podpis

Název diplomové práce:

Optimalizace výroby s využitím pokročilých metod pro podporu rozhodování

Abstrakt:

Práce se bude zabývat problematikou procesního řízení ve výrobní společnosti s využitím pokročilých metod pro podporu manažerského rozhodování. Konkrétně zde bude řešena optimalizace výroby za pomoci metody genetických algoritmů a lineárního programování. Cílem práce je vytvořit suboptimální řešení rozvržení výrobního plánu tak, aby došlo k minimalizaci celkového času pro zhotovení všech zadaných zakázek. Řešitelka zde bude navržený algoritmus aplikovat ve výrobní společnosti XY.

Klíčová slova:

Optimalizace výroby, řazení zakázek, rozvrhovací úloha, genetické algoritmy, lineární programování

Poděkování:

Tímto bych chtěla poděkovat mému vedoucímu práce doc. Ing. Vladislavu Bínovi, Ph.D. za pomoc při tvorbě zejména praktické části práce. Dále bych ráda poděkovala mým přátelům a rodině za podporu a konzultace.

Obsah

Seznam obrázků	11
Seznam tabulek	12
Úvod	13
1 Procesní řízení	15
1.1 Proces	15
1.1.1 Komponenty procesu	15
1.1.2 Životní cyklus procesu	17
1.1.3 Klasifikace procesů	19
1.2 Procesní přístup	20
1.2.1 Rozdíl mezi funkčním a procesním řízením	21
1.3 Business proces reengineering (BPR)	22
2 Výroba a její řízení	25
2.1 Logisticky řízený podnik	26
2.2 Zakázková výroba (Job-shop production)	27
2.3 Optimalizace výroby	28
2.3.1 Přínosy optimalizace výroby	28
2.3.2 Výrobní náklady	29
3 Rozvrhování výroby	30
3.1 Rozvrhování zakázkové výroby	30
3.2 Prioritní pravidla	34
3.2.1 First come first served (FCFS)	34
3.2.2 Shortest job next (SJN)	35
3.2.3 Priority based scheduling (PBS)	37
3.2.4 Round-Robin scheduling (RBS)	38
3.2.5 Porovnání přístupů	39
4 Metody řešení	40
4.1 Přesné metody	40
4.2 Heuristické metody	41
5 Metodika práce	42
6 Genetické algoritmy	44
6.1 Historie	44
6.2 Terminologie	44
6.3 Základní funkce a principy	45
6.3.1 Vytvoření počáteční generace	46
6.3.2 Výběrové metody (selekce)	47
6.3.3 Operátory křížení	50
6.3.4 Operátory mutace	52

6.3.5 Reprodukce	53
6.4 Praktické využití GA	53
7 Lineární programování	54
7.1 Fáze aplikace lineárního programování.....	55
7.2 Ekonomický model LP.....	56
7.2.1 Vzorový příklad-ekonomický model.....	57
7.3 Matematický model LP	57
7.3.1 Vzorový příklad-matematický model.....	58
7.4 Simplexová metoda	59
7.4.1 Jednofázová simplexová metoda	61
8 Popis společnosti XY	65
8.1 Současná situace ve společnosti XY	65
8.2 Zdrojová data	67
8.2.1 Úprava dat.....	67
9 Řešení pomocí genetických algoritmů	72
9.1 Ukázkový příklad.....	72
9.2 Výsledné řešení	74
9.3 Interpretace výsledků	75
9.4 Kód pro zadaný případ	76
10 Řešení pomocí lineárního programování	79
10.1 Popis kódu	79
10.2 Výsledné řešení	82
10.3 Interpretace výsledků.....	83
11 Porovnání výsledků.....	86
Závěr	87
Přílohy	96
Příloha 1-Původní zdrojová data	96

Seznam obrázků

Obrázek 1: Komponenty procesu	17
Obrázek 2: Životní cyklus procesu	18
Obrázek 3: Zakázková výroba-schéma.....	27
Obrázek 4: Tabulka zadání problému 3x3.....	31
Obrázek 5: Ganttův diagram problému 3x3.....	32
Obrázek 6: Disjunktivní graf	33
Obrázek 7: FCFS nákres	35
Obrázek 8: SJN nákres.....	36
Obrázek 9: PBS nákres.....	37
Obrázek 10: RBS nákres.....	38
Obrázek 11: Základní princip genetických algoritmů	45
Obrázek 12: Ruletový výběr.....	48
Obrázek 13: Fáze při aplikaci operačního výzkumu	56
Obrázek 14: Hrubé schéma simplexové metody	60
Obrázek 15: Schéma zakázek ve výrobě.....	71
Obrázek 16: Ganttův diagram LP.....	75
Obrázek 17: Ganttův diagram- základní řešení GA.....	83
Obrázek 18: Rozpracovaný Ganttův diagram	84
Obrázek 19: Detailní Ganttův diagram-metoda GA	85

Seznam tabulek

Tabulka 1: Klasifikace procesů	19
Tabulka 2: Rozdíly mezi funkčním a procesním řízením	21
Tabulka 3: Základní charakteristiky TQM vs. Reengineering	23
Tabulka 4: Výrobní proces v podniku	26
Tabulka 5: Klasifikace algoritmů	34
Tabulka 6: FCFS praktický příklad-zadání.....	35
Tabulka 7: FCFS praktický příklad-řešení	35
Tabulka 8: SJN praktický příklad-zadání.....	36
Tabulka 9: SJN praktický příklad-řešení	36
Tabulka 10: PBS praktický příklad-zadání.....	37
Tabulka 11: PBS praktický příklad-řešení.....	38
Tabulka 12: RBS praktický příklad-zadání.....	38
Tabulka 13: RBS praktický příklad-řešení.....	39
Tabulka 14: Porovnání metod.....	39
Tabulka 15: Rozdělení heuristických metod	41
Tabulka 16: Jednoduché zakódování populace	46
Tabulka 17: Ohodnocení funkcí fitness.....	46
Tabulka 18: Turnajový výběr	49
Tabulka 19: Jednobodové křížení.....	50
Tabulka 20: Dvoubodové křížení.....	51
Tabulka 21: Vícebodové křížení.....	51
Tabulka 22: Uniformní křížení.....	51
Tabulka 23: Binární mutace.....	52
Tabulka 24: Prvky ekonomického a matematického modelu	54
Tabulka 25: Výchozí simplexová tabulka.....	62
Tabulka 26: Výchozí simplexová tabulka s vyznačeným klíčovým sloupcem a řádkem	63
Tabulka 27: První krok výpočtu simplexovou metodou	63
Tabulka 28: Optimální řešení úlohy lineárního programování.....	64
Tabulka 29: Výpočet pracovních hodin	65
Tabulka 30: Stávající výrobní plán.....	66
Tabulka 31: Výchozí datový soubor	68
Tabulka 32: Pokusy řešení metodou LP	75

Úvod

Z důvodu dynamicky se měnícího prostředí je třeba v organizacích působících v různých odvětvích zavádět základy procesního řízení. Zavedením procesního řízení je organizace schopna flexibilně reagovat na turbulentní prostředí a rychle se měnící požadavky zákazníků. Prostřednictvím procesního řízení a reengineeringu základních operací se zvyšuje i efektivita v organizacích, zejména pak ve výrobních společnostech.

Cílem diplomové práce bylo zabývat se procesním řízením ve výrobní organizaci a jeho přínosy pro zvýšení efektivity organizace. Při psaní této práce budu spolupracovat s nejmenovanou společností XY, která mi poskytne potřebná data a prostředí pro praktickou aplikaci zjištěných poznatků. Stěžejní částí práce je zaměření na optimalizaci procesů ve výrobě. Tato optimalizace se skládá z řazení zakázek, s cílem zkrátit čas zakázky strávený ve výrobním procesu a tím i celkový čas na zhotovení všech zakázek.

V práci bude provedeno porovnání dvou metod použitých k seřazení zakázek přicházejících do výroby. Jako první metoda byly zvoleny genetické algoritmy. Pro druhou bylo využito lineární programování.

V diplomové práci jsou kladeny následující výzkumné otázky:

- *Která metoda je pro řešení rozvrhovací úlohy vhodnější?*
- *Lze nalézt takový způsob řazení zakázek, který bude ekonomicky výhodnější než řazení stávající?*
- *Jak správně nastavit algoritmus pro seřazení zakázek ve výrobě?*

Hlavním cílem práce je navrhnout a porovnat metody využitelné pro optimalizaci výrobního plánu ve společnosti. Naprogramovat algoritmus pro řešení rozvrhovací úlohy ve výrobní společnosti, aplikovat tento algoritmus na data z konkrétní výrobní společnosti a poté uvést doporučení pro řešení tohoto druhu optimalizace výroby pro firmu XY.

V první kapitole práce uvedu čtenáře do problematiky procesního řízení, vysvětlím důležité pojmy v tomto oboru, a také nové trendy a metody. Více se zaměřím na aplikaci reengineeringu a optimalizace výroby, jaké procesy a jejich části je třeba optimalizovat a jakých metodik se k tomu využívá. Hlavním předmětem práce je optimalizace řazení zakázek ve výrobě, kterou se budu podrobněji zabývat v další podkapitole první části práce.

Dále v metodice práce navrhnou metody vhodné pro zadaný praktický případ, popíši základní rysy těchto metod a posléze budu argumentovat jejich využití. Pro

aplikační část práce volím metodu genetických algoritmů, kterou porovnáím s poněkud známější metodou lineárního programování. Blíže zde popíši jejich funkce, základní princip, a také jejich využití v praktických úlohách. Posledním tématem teoretické oblasti práce bude představení společnosti XY, optimalizace jejichž výrobních procesů je základem pro praktickou část práce.

Jako první se v praktické části práce budu zabývat metodou genetických algoritmů, nastavením algoritmu a následnou aplikací na datový soubor získaný od společnosti XY. K aplikaci algoritmu na konkrétně zadaná data je třeba využít vhodného softwaru, v mé práci pracuji s programovacím jazykem R. Výsledky aplikace genetických algoritmů budou, pro lepší prezentaci, zpracovány do formy Ganttova diagramu. Dále přistoupím k aplikaci metody lineárního programování, také zde uvedu kód pro generování možných řešení zadané úlohy a výsledky opět převedu do Ganttova diagramu.

Poslední část bude věnována zobecnění výsledků práce a porovnání dvou použitých metod. Mým předpokladem je, že metoda genetických algoritmů bude pro řešení úlohy rozvrhovacího problému vhodnější než metoda lineárního programování. Při tomto úsudku vycházím z faktu, že genetické algoritmy jsou vysoce sofistikovaná metoda, určená pro řešení složitých optimalizačních problémů. Metoda lineárního programování je sice také vhodná pro řešení optimalizačních úloh, avšak s touto metodou jsem se setkala již při bakalářském studiu, usuzuji tak, že je jednodušší, a je tedy vhodná spíše pro řešení základních jednoduchých problémů.

1 Procesní řízení

Procesní řízení, neboli Business process management (BPM) je soubor činností týkajících se plánování a sledování výkonnosti firemních procesů, zejména těch realizačních). Jinými slovy je procesní řízení takový způsob řízení procesů v organizaci, který zdůrazňuje opakované procesy a jejich průběh napříč celou organizací. Cílem BPM je zajištění rozměrů podnikání ve všech jejich vazbách a dynamice změn s využitím možností vývoje informační technologie. BPM navazuje na koncept reengineeringu podnikových procesů, který byl popsán v knize *Reengineering the corporation: A Manifesto For Business Revolution* z roku 1993, kterou napsali průkopníci v tomto odvětví Michael Hammer a James Champy. Cílem procesního řízení je rozvíjet a optimalizovat chod organizace tak, aby efektivně, účelně a hospodárně reagovala na požadavky zákazníka (Grasseová, 2008).

1.1 Proces

Proces je soubor činností, který vyžaduje jeden nebo více druhů vstupů, a tvoří výstup, který má pro zákazníka hodnotu. (Hammer a Champy, 1996). Další z nejvyužívanějších definic je: „Proces je strukturovaný a měřitelný set aktivit, vytvořených pro výrobu speciálního výstupu pro určitého zákazníka nebo určitý trh. Klade se velký důraz na to, jak je práce prováděna v rámci organizace“ (Davenport a Stoddart 1994). Dle Grasseové je proces soubor vzájemně souvisejících nebo vzájemně působících činností, které dávají přidanou hodnotu vstupům, při využití zdrojů, a přeměňují je na výstupy, které mají svého zákazníka. Proces je možné chápat také jako organizovanou skupinu vzájemně souvisejících činností (subprocesů), které procházejí jedním nebo více organizačními útvary či jednou (podnikový proces) nebo více spolupracujícími organizacemi (mezipodnikový proces), které spotřebovávají materiální, lidské, finanční, a informační vstupy a jejichž výstupem je produkt, který má hodnotu pro externího nebo interního zákazníka (Šmída, 2007).

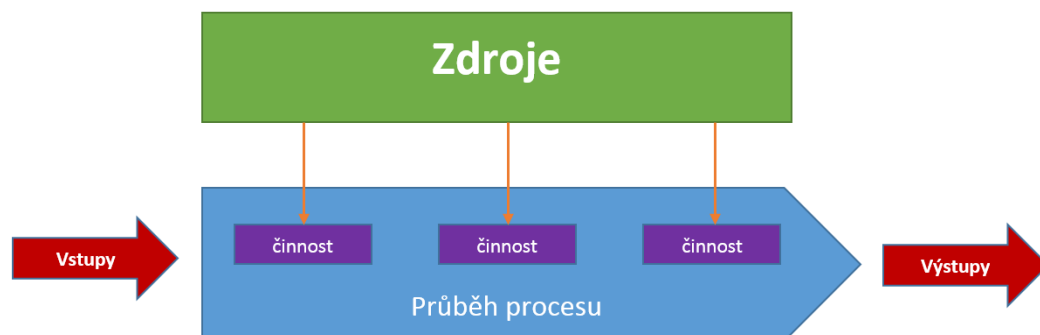
Definici procesu existuje celá řada, z obecného hlediska je proces sled činností a vazeb mezi nimi. Na začátku každého procesu jsou vždy vstupy, důležitým výstupem procesu je přidaná hodnota pro zákazníka.

1.1.1 Komponenty procesu

Každý proces se skládá z komponentů, které do něj vstupují a ovlivňují tak jeho průběh, a také jeho výstupy, názorně je schéma procesu ukázáno na obrázku (Obrázek 1). Mezi základní komponenty procesu patří:

- **Události** – určité podmínky, které musí být splněny, aby mohlo dojít k provedení procesu. Je to něco, co se děje bez ohledu na to, co je účelem daného procesu. Můžeme je chápat jako určité efekty, které jsou zjištěny po tom, co nastane nějaká příčina. Každý proces začíná a končí s určitou událostí.
- **Úkoly** – úkoly jsou nejmenší možné jednotky (základní jednotka), na které může být aktivita rozdělena. Business proces popisuje různé činnosti a vzájemný vztah mezi nimi. Důležitým poznatkem je, že vazby mezi úkoly mohou být mnohdy důležitější než úkoly samotné. Platí zde tzv. synergický efekt, celek má větší výslednou hodnotu než pouhý souhrn všech jeho částí. Největší problém BPM je ten, že většina jeho realizátorů není schopna pochopit jeho systémové hledisko. Zaměstnanci jedné firmy mají mnohdy protichůdné cíle (Jablonský, 2007).
- **Rozhodnutí** – mohou existovat určitá rozhodnutí, která jsou přijímána jako součást procesu. Ponechání odpovědnosti na osobách zúčastněných v procesu může mít mnohdy i negativní důsledky. Je totiž pravděpodobné, že jestliže neexistují jasné pokyny, pak se rozhodnutí přijatá různými lidmi mohou lišit, tím vzniká hrozba vzniku nekonzistentních zážitků pro zákazníky, což má vliv na kvalitu poskytované služby.
- **Vstupy** – dokud jsou do procesu vkládány vstupy, proces nemůže začít plně fungovat. Správné vstupy jsou pro proces velmi důležité. Vložení špatných vstupů je pro proces negativní a neefektivní. Níže popisují některé vstupy, které jsou pro správnou funkci procesu nezbytné (Jablonský, 2007).
 - **Lidé** – každý proces práci lidí a jejich odpovídající schopnosti a dovednosti. To je také důvod, proč je rozdělování úkolů a jejich delegování tak důležité. V procesu řízení organizace mohou méně kvalifikovaní jedinci vykonávat běžné, jednodušší úkoly, zatímco odborníci a více kvalifikovaní lidé se mohou podílet na řešení složitých a nevšedních problémů a úkolů (Chase a Aquilano, 1995). Soulad v dovednostech pracovní síly a požadavků jim přidělených snižuje náklady a zvyšuje efektivitu procesu.
 - **Materiál** – Suroviny neboli materiál pro výrobu musí být vložen včas a s co možná nejnižšími náklady. Existuje mnoho organizací, které mají nákupní a skladovací procesy postaveny jako jejich klíčovou kompetenci.
 - **Informace** – Správné odpovídající informace musí být dány k dispozici všem subjektům v procesu. Pracovníci musí mít určité schopnosti a dovednosti a musí být dobře obeznámeni s postupem. Manažer musí dostávat kontinuální zpětnou vazbu, aby bylo zajištěno, že výroba bude dosahovat cílů, jak bylo naplánováno.

- **Výstupy** – Výstupy z procesu musí být průběžně monitorovány. Toto později pomůže při měření účinnosti a efektivity celého procesu a poté při navrhování změn v procesu, které budou vyžadovány (vom Brocke a Rosemann, 2015).



Obrázek 1: Komponenty procesu

Vlastní zpracování, zdroj: Pešková, 2016

1.1.2 Životní cyklus procesu

Na životní cyklus procesu opět existuje několik pohledů. Dle M. Tůma a J. Basl (2002) má proces pouze 3 hlavní etapy – Návrh procesu, Implementaci procesu a Průběžnou optimalizaci procesu. Z mého hlediska je však výstižnější názor, že proces prochází následujícími pěti etapami, které jsou zobrazeny i na obrázku níže (Obrázek 2):

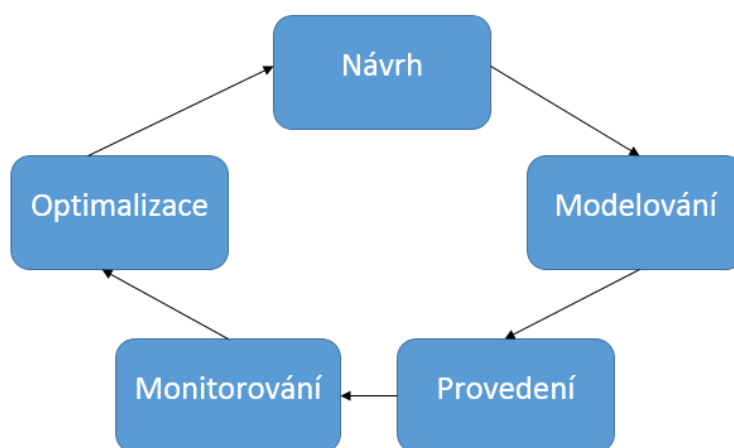
Návrh – design nebo také vytvoření procesu. V první části životního cyklu probíhá jak identifikace a posouzení již existujících procesů, tak vytvoření nových, budoucích procesů. Zaměřujeme se zde na reprezentaci průběhu a toku procesů, faktorů v něm vystupujících, výstrah a oznámení a standardních operačních procedur. Dále zde probíhá domluva o úrovni a kvalitě služeb a na závěr definice stěžejních úkolů v procesu a předávacích mechanismů. Cílem tohoto kroku je zjištění, zda je připravený dostatečný teoretický podklad pro vznik nového procesu. Stávající proces a návrh nového procesu je nutné synchronizovat tak, aby nedošlo k pozdějšímu významnému výpadku v systému.

Modelování – V této fázi je důležité získat dostatek podrobností o procesu a koncepčně pochopit, jak proces funguje. Nepoužívají se zde dlouhé historické obchodní analýzy, ale naopak se zaměřujeme na snadno použitelné a jednoduché technické analýzy, které jsou pro dynamické prostředí procesního managementu vhodnější. Můžeme zde spustit tzv. analýzu *Co-když?* (What-if), tato metoda se řadí mezi jednoduché analytické techniky, pomocí které zjišťujeme možné důsledky našeho jednání a scénáře budoucích stavů procesů a následně uvádíme opatření proti těmto dopadům.

Provedení (uskutečnění, execution) – V této fázi probíhá samotná implementace, neboli realizace procesů. Jedním ze způsobů, jak automatizovat procesy je vyvinout nebo zakoupit aplikaci, která sama vykonává určité kroky procesu. V praxi však ne vždy tyto aplikace vykonávají kroky přesně a úplně. Další možností je využití softwaru v kombinaci s lidským zásahem, tento způsob je však složitější a náročnější na dokumentaci.

Monitorování – fáze monitoringu zahrnuje sledování jednotlivých procesů tak, že stav procesu je zjevný a transparentní. Poté můžeme vytvářet různé statistiky pro zhodnocení výkonu procesu. Kromě toho mohou být tyto informace použity pro práci se zákazníky a s dodavateli. Stupeň sledování závisí na tom, jaké informace firma chce vyhodnocovat a analyzovat, a také do jaké míry chce firma získávat informace v reálném čase nebo jestli jí stačí tzv. ad-hoc informace a data. Zavádí se zde také pojem Process mining jako soubor metod a nástrojů vztahujících se k procesu a jeho sledování. Cílem tohoto postupu je analyzovat protokoly událostí extrahované prostřednictvím monitorování procesů.

Optimalizace – Optimalizace procesu zahrnuje získávání informací o výkonu procesu od fáze modelování až po fázi monitoringu. Dále určení potencionálních překážek a potenciálních příležitostí pro úsporu nákladů nebo pro jakékoliv jiné zlepšení. Pomyslný kruh cyklu procesu se uzavírá tak, že zjištěné možnosti pro zlepšení procesu se následně použijí ve fázi designu procesu. Nástroje process miningu jsou schopny odhalit kritické činnosti a úzká místa v procesu (Process approach, 1999).



Obrázek 2: Životní cyklus procesu

Vlastní zpracování, zdroj: ExpertBA, 2012

1.1.3 Klasifikace procesů

Existuje celá škála různých procesů, které se liší svým obsahem, dobou existence, významem, účelem nebo třeba frekvencí opakování. Základní dělení procesů je podle jejich významu. Takto procesy rozdělujeme na **hlavní**, **řídící** a **podpůrné** (Motalík, 2009). Hlavní procesy (obchodní) vytvářejí hodnotu v podobě výrobku nebo služby pro zákazníka, jsou tedy hlavními procesy, které přispívají k naplnění poslání organizace. Řídící procesy (manažerské) zajišťují rozvoj a řízení výkonu organizace a zajišťují tak její integritu a fungování. Podpůrné procesy nakonec zajišťují samotný chod organizace. Toto dělení procesů, zobrazené i v tabulce (Tabulka 1) je zároveň v praxi nejpoužívanější. Nejen že je přehledné a jednoduché, ale hlavně nám poskytuje důležité informace o procesu a také nám napovídá, jak by měl být proces řízen (Svobodová, 2008). Také jasně ukazuje na význam jednotlivých procesů, a tím pomáhá stanovit priority procesů pro reengineering.

Tabulka 1: Klasifikace procesů

Typ procesu	Způsob, jakým má být řízen	Charakteristika procesu			
		Přidává hodnotu?	Probíhá napříč organizací?	Má externí zákazníky?	Generuje zisk?
Hlavní	Výkonově	ANO	ANO	ANO	ANO
Řídící	Nákladově	NE	ANO	NE	NE
Podpůrný	Výkonově, outsourcing	ANO	NE	NE	NE

Vlastní zpracování, zdroj: Motalík, 2009

Dále dělíme procesy podle jejich struktury na **datové** a **znalostní**. U datových procesů je seznam a pořadí činností přesně popsán a toto pořadí již nelze měnit. Naopak u znalostních procesů seznam a pořadí činností není jasně vymezen a je tedy možné jej měnit podle vzniklé situace, zde se tedy jedná o tvůrčí a znalostní procesy (Motalík, 2009). Dále rozdělujeme procesy podle doby jejich trvání, a sice na **trvalé** a **dočasné**, kde dočasné procesy, nebo také jednorázové procesy, jsou časově omezené a mají z pravidla charakter procesu. Procesy také můžeme dělit podle frekvence jejich opakování na **procesy s vysokou mírou opakovatelnosti**, které nastávají minimálně jednou za rok a **procesy s nízkou opakovatelností**. Na závěr bych zde uvedla dělení procesů ze strategického hlediska, které je všeobecně používané a velmi dobře známe, není tedy nutné je zde více rozvádět. Z tohoto hlediska dělíme procesy na **strategické**, **taktické** a **operativní** (Zelená, 2015).

1.2 Procesní přístup

Základní charakteristikou procesního přístupu k řízení je schopnost reagovat na dynamické prostředí a rychle a neustále se měnící požadavky zákazníků. Jedná se o manažerskou disciplínu, která se opírá o uchopení struktur firmy, jejich architektury a řízení prostřednictvím podnikového modelu (Enterprise model). Enterprise model zachycuje tyto základní rozměry podnikání:

- cíle podnikání,
- hodnotvorné procesy,
- organizační, znalostní a informační infrastruktury procesu a podpůrné technologie.

Klíčovým faktorem úspěchu je důslednost zejména v nasazení a prosazování procesního přístupu za jednoznačné a trvalé podpory vrcholového managementu (Brychta, 2008).

Procesní řízení lze označit za souhrn systémů, postupů, metod a nástrojů pro trvalé zajištění maximální výkonnosti a neustálého zlepšování podnikových procesů. Cílem procesního řízení je rozvíjet a optimalizovat chod podniku tak, aby mohl efektivně, účelně a hospodárně reagovat na měnící se požadavky zákazníků.

Procesní přístup je charakterizován základními principy procesního řízení, jimiž jsou:

- **Integrace a komprese prací** – jednotlivé práce (činnosti) se integrují do logických celků tak, aby je procesní tým, který se zaměřuje na přidanou hodnotu pro zákazníka, byl schopen obsáhnout. Tak zvaná komprese těchto prací znamená napřimování procesů a vede k jejich přeprojektování. Jde o vyloučení nadbytečných činností, doplnění činností chybějících a optimalizaci neefektivně prováděných činností.
- **Delinearizace prací** – práce je vykonávána v přirozeném sledu.
- **Uplatnění týmové práce** – vyskytují se zde tzv. autonomní týmy, které zajišťují procesy, tyto týmy disponují dostatečnými pravomocemi a jejich motivace je přímo svázána s přidanou hodnotou pro zákazníka.
- **Odpovědnost za proces** – vlastník procesu je odpovědný za proces a za jeho efektivnost v dlouhodobém horizontu (je nutná znalost zákazníka a jeho potřeb).
- **Znalost procesů** – každá organizace musí detailně znát své procesy a jejich vstupy a výstupy. Dále způsob, jak jsou vstupy přeměňovány na výstupy a které zdroje jsou při tomto procesu spotřebovávány a v jakém množství.
- **Verifikace činností pro přeměnu vstupů na výstupy** – činnosti (úkoly) prováděné v rámci procesu jsou důkladně zmapovány a parametrizovány. Ze zjištěných

informací jsou později vytvořeny výkonnostní charakteristiky. Při přeměně vstupů na výstupy je třeba dostatečně vymezit role jednotlivých pracovníků.

- **Monitorování, měření a neustálé zlepšování** – vlastník procesu má k dispozici výkonnostní ukazatele, které vypovídají o účinnosti a efektivnosti procesů. Na základě těchto ukazatelů navrhuje a provádějí změny v procesech, čímž dochází k celkové optimalizaci procesů (Grasseová, Dubec a Horák, 2008).

1.2.1 Rozdíl mezi funkčním a procesním řízením

Procesní řízení je zaměřeno nejen na výsledek práce, ale i na postup a jeho dosažení, zatímco u funkčního řízení se klade největší důraz na organizační dělení společnosti dle dovedností pracovníků a tvorbu struktury organizace. V rámci procesního řízení organizace je celý systém řízen potřebami zákazníka, nejčastěji se tak děje formou řízení produktového portfolia. U procesního řízení obvykle dochází ke zlepšení prostřednictvím optimalizace procesů a zjednodušením toku práce. U funkčního řízení jsou organizační jednotky přesně vymezené a známé, avšak procesy zde nejsou jasně zmapované a definované. Z toho vychází, že pracovníci uvažují pouze o jednotlivých činnostech místo o procesu jako celku a samotné procesy zůstávají neřízeny. Základní rozdíly mezi funkčním a procesním řízením v organizaci jsou zachyceny v následující tabulce (Tabulka 2) (Motalík, 2009).

Tabulka 2: Rozdíly mezi funkčním a procesním řízením

Kritérium	Funkční řízení	Procesní řízení
Základní princip	Dělbá práce	Integrace činností
Základní stavební jednotka	Dílčí operace	Proces
Zájem je soustředěn na...	Činnost	Výsledek
Charakter výroby	Hromadná	Variantnost
Základní aktivum	Kapitál	Znalosti
Předpoklad úspěchu	Objem, rychlost	Pružnost
Podnik jako systém	Koordinace oddělených prvků	Snaha o synergický efekt
Ukazatelé úspěšnosti	Ekonomické ukazatele	Přidaná hodnota pro zákazníka
Organizační struktura	Strmá pyramida	Horizontální, plochá
Řízení	Hierarchické	Napříč útvary

Pravomoci a odpovědnost	Vymezená-za operaci nebo úsek	Za proces
Vztah k podřízeným	Kontrola, přikazování, tvrdé prvky	Koučování, měkké prvky
Ukazatele podniku	Ekonomická analýza	Analýza procesů
Orientace	Důsledky	Příčiny
Hlavní funkce podniku	Výroba	Marketing
Okolní prostředí	Ekonomika orientovaná na rozsah	Znalostní ekonomika
Management řídí	Jednotlivce	Týmy
Management	Operační	Procesní
Vnitropodnikové prostředí	Konkurence mezi funkcemi	Spolupráce
Charakter práce	Specializace	Integrace
Kvalifikace	Nenáročná	Náročná na kvalifikaci
Motivace	Splnění ukazatelů spojených s činností	Hodnotová metrika zaměřená na proces
Komunikace	Lineárně vertikální	Horizontální
Lidé	Industriální člověk	Znalostní člověk
Myšlení	Deduktivní	Induktivní

Vlastní zpracování, zdroj: Zelená, 2015

1.3 Business proces reengineering (BPR)

Z důvodu globalizace ekonomik a trhů po celém světě dochází v obchodním prostředí k dynamickým změnám a k neustále se zvyšujícímu konkurenčnímu boji. Konkurence sílí hlavně kvůli odstranění bariér obchodu a značnému technologickému pokroku. Všechny tyto změny zapříčinily potřebu organizační transformace, při které se mění nejen organizační struktura společnosti a její organizační prostředí a kultura, ale mění se celkově všechny procesy (Davenport, Thomas a Stoddard, 1994).

Pojem Business proces reengineering se začal objevovat na území Ameriky během let 1980 až 1990, nejprve v soukromém sektoru a později pronikl i do sektoru veřejného. Úspěch BPR je často přisuzován skutečnosti, že staré organizační způsoby již v současných podmínkách nefungují a z dlouhodobého hlediska se od nich upouští (Hammer a Champy, 1993). Reengineering je specifický a zcela odlišný přístup, který

je charakteristický tvorbou nových a efektivních procesů v organizaci a jeho základní charakteristikou je to že nebere v úvahu minulost. BPR je definován takto: „přehodnocení a redesign základních procesů v organizaci za účelem dosažení dramatického zlepšení v kritických výkonových měřítkách, jakými jsou náklady, kvalita, servis a rychlost“ (Hammer a Champy, 1993). Jinými slovy je business proces reengineering dynamický přístup k zavádění procesního řízení, který je založený na radikální změně organizační struktury. Reengineering je tedy společně s Total Quality Managementem (TQM) jednou z metod procesního řízení (Zigiaris, 2000), porovnání těchto dvou metod je vidět níže (Tabulka 3).

Tabulka 3: Základní charakteristiky TQM vs. Reengineering

Total Quality Management	Reengineering
Leadership	Procesní řízení
Orientace na zákazníka	Týmová práce
Zaměření na trvalé zlepšování	Změna úlohy manažerů
Důraz na priority a prevenci	Změna organizační struktury
Procesní přístup	Změna chápání zákazníka
Bezvadnost samozřejmostí	Aktivní účast všech zaměstnanců
	Tréninkové a školicí programy
	Zpětná vazba
	Vazby na jiné systémy a programy
	Náročnost na zdroje (finanční, lidské)

Při implementaci business proces reengineering přístupu v organizaci se realizační tým zaměřuje na různé cíle:

- **Zaměření na zákazníka:** je třeba se soustředit na procesy v rámci zákaznického servisu, prostřednictvím toho jsou eliminovány stížnosti a připomínky od zákazníků.
- **Rychlost:** Jde o dramatické zkrácení času potřebného pro zhotovení klíčových procesů u zakázky. Pro příklad, jestliže proces před zavedením BPR zabral průměrně 5 hodin, po zavedení BPR by tento čas měl být zkrácen až na pouhou půlhodinu (Zigiaris, 2000).
- **Komprese:** Snížení hlavních nákladových položek prostřednictvím hodnotového řetězce. Organizací procesů společnost vytváří a udržuje transparentnost. Příkladem je nákup velkého množství surového materiálu se slevou 50 %, který je spojen

s 11 kontrolami napříč celou organizační strukturou od cash flow a skladování, přes plánování výroby, až po marketing. Tyto kontroly se stávají snadno proveditelnými díky zavedení tzv. cross-functional týmů, optimalizaci rozhodovacích procesů a snižováním provozních nákladů (Motalík, 2009).

- **Flexibilita:** Adaptivní procesy a struktury mění podmínky a konkurenci. Tím, že se organizace snaží přiblížit zákazníkovi může vytvořit mechanismy na zvýšení povědomí a pro rychlé odhalení slabých míst. To všechno pomáhá k lepší a rychlejší reakci a adaptabilitě na neustále se měnící požadavky trhu.
- **Kvalita:** Je třeba se zaměřit hlavně na vynikající servis a přidanou hodnotu pro zákazníka. Úroveň kvality je vždy pečlivě sledována a kontrolována v několika procesech. Úroveň kvality by neměla být ve všech případech závislá na osobě, která právě obsluhuje zákazníka.
- **Inovace:** Vedení lidí prostřednictvím kreativních změn poskytuje organizaci určitou konkurenční výhodu.
- **Produktivita:** Cílem BPR je zejména výrazně zvýšit efektivitu a efektivnost v organizaci (Heřman, 2001).

2 Výroba a její řízení

Pod pojem plánování výroby zahrnujeme celý proces komplexního zpracování zakázek od marketingu po sledování a řízení výrobního procesu. Jedním z nejdůležitějších aspektů procesu plánování a řízení výroby je stanovení pohybu materiálu v celém výrobním procesu nebo jeho části. Mezi základní principy, uplatňované při organizaci pohybu materiálu ve firmě, řadíme principy push a pull. U principu push je materiál na jednotlivá pracoviště dodáván podle předem stanoveného plánu bez ohledu na skutečnou potřebu. Na pracovišti se pak materiál může hromadit a vytvářet se tak zbytečné zásoby (Heřman, 2001). Naproti tomu u principu pull pracoviště odebírá materiál na základě okamžité potřeby. Materiál se tak ihned zpracovává a neskládá se na jednotlivých pracovištích.

Pro plánování výroby se v dnešní době používá tzv. MRP (Material Requirements Planning) systémů. Tyto počítačově podporované systémy jsou zaměřeny jednak na minimalizaci nákladů na materiálové požadavky, tak na nákupní, finanční či marketingové aspekty nebo například na predikci vývoje požadavků zákazníků. Nespornými výhodami implementace je pozitivní vliv na finanční výsledky podniku, zlepšení výkonu a řízení výroby, snížení výrobních nákladů nebo zajištění vyšší spolehlivosti výroby. Naopak jsou zde také četné nevýhody, jako například riziko výpadku nebo zpomalení výroby při nepředvídatelných problémech s dodávkami, je tedy nutno udržovat pojistnou zásobu. Největší nevýhoda je však vnímána v tom, že MRP systémy jsou ve většině případů kupovány jako standardizované softwarové balíky, které je zpravidla nutno komplikovaně přizpůsobovat konkrétním podmínkám ve společnosti (Drdová, 2007).

Řízení výroby je zaměřeno na dosažení optimálního fungování výrobních systémů s ohledem na vytyčené cíle. V řízení výroby se jedná o věcné, prostorové a časové sladění a také případnou koordinaci činitelů účastnících se výrobních procesů nebo výrobní procesy ovlivňujících: pracovníků podílejících se na výrobě, provozních prostor, nezbytných výrobních a dopravních zařízení, surovin, polotovarů, energií, rozpracovaných výrobků, finančních prostředků, informací a samozřejmě také odpadů.

Řízení výrobních procesů je nedílnou funkční součástí každé organizace. Prosperita podniku je založena na spolupráci a řízení jednotlivých funkčních oblastí, a to především marketingu, financí a výroby. Řízení jednotlivých oblastí by mělo směřovat ke společnému cíli, přestože krátkodobé cíle těchto jednotlivých oblastí bývají konkurenční.

2.1 Logisticky řízený podnik

Logisticky řízený podnik se vyznačuje silnou orientací na zákazníka, na přidanou hodnotu a totální kvalitu výrobků. V takových podnicích se uplatňují především měkké faktory řízení, jako jsou sociokulturní regulátory, vnitřní motivace, sebekontrola a také podniková kultura. Kontrolní systém se zde soustřeďuje na kontrolu výrobního procesu jako takového, zatímco u tradičně řízeného podniku se kontrolují především lidé, zda se řídí danými pokyny. U logisticky řízených podniků se také dosahuje výrazně nižšího procenta vad u vyprodukovaných výrobků (Heřman, 2001). Tabulka 4 ukazuje rozdíly ve výrobním procesu u tradičního podniku a logisticky řízeného podniku.

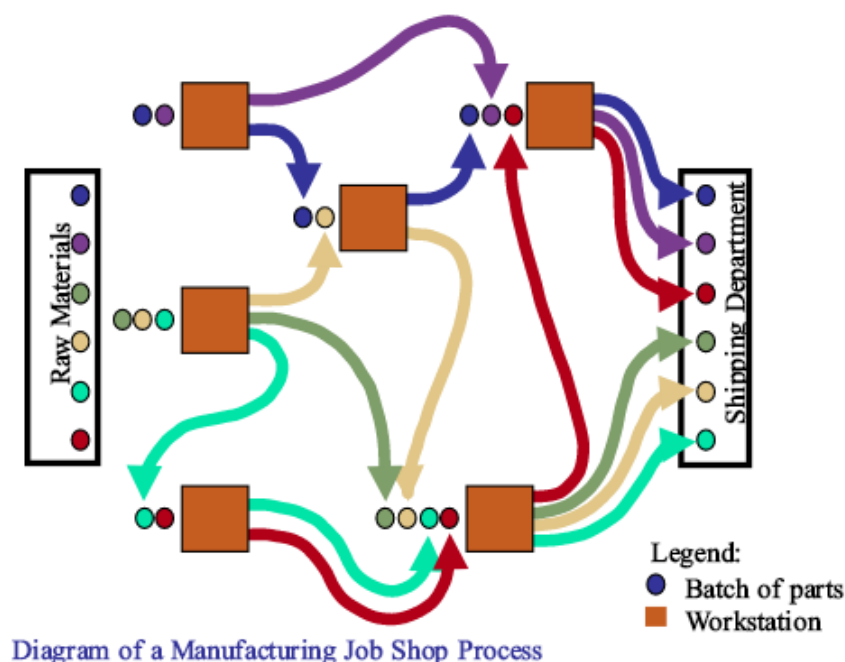
Tabulka 4: Výrobní proces v podniku

Výrobní proces	Tradiční podnik	Logisticky řízený podnik
výrobní cyklus	Dlouhý	Krátký
výrobní série	Velké	Malé
rozmístění výrobního zařízení	Podle procesů, v dílnách	Podle výrobků, v buňkách
práce dělníků	Dekvalifikována k dosažení nízkých mzdových nákladů	Jejich kvalifikace se zvyšuje a využívá k zlepšení výrobního procesu
zainteresovanost pracovníků	Nízká	Vysoká
automatizace	Automatizován neracionalizovaný proces; ostrůvky automatizace, izolované roboty	Automatizace po zjednodušení výrobního procesu; integrované systémy

U logisticky řízeného podniku je základem úzká spolupráce nebo dokonce partnerství s dodavateli. Důležité je samozřejmě také srovnání cílů a potřeb podniku s potřebami zákazníků a také s konkurenty. Současným trendem při řízení a optimalizaci výroby je plynulý přechod od tradičně řízeného podniku k logisticky řízenému podniku, tak, aby bylo dosaženo tzv. učící se organizace.

2.2 Zakázková výroba (Job-shop production)

Zakázková výroba se vyznačuje tím, že výrobky jsou zde vyráběny ve skupinách (zakázkách), oproti tomu u sériového typu výroby jsou produkty zpracovávány v souvislém proudu. Je to typ výrobního procesu, ve kterém jsou vyráběny malé skupiny výrobků a jsou přizpůsobovány na přání zákazníka (Drdová, 2007). V toku výrobního procesu většina výrobků vyžaduje unikátní nastavení strojů a pečlivé rozvrhování jednotlivých kroků. Tento typ výroby je většinou používán u takových výrobních firem, které vyrábějí komponenty a suplementy pro jiné podniky. Tento typ výroby je využíván v rozmanité škále podniků, a to od strojírenských firem, přes prodejnu barev až po tiskárny. Všechny tyto podniky mají však společné to, že vyrábějí výrobky po malých sériích (zakázkách), značně se přizpůsobují přání zákazníků a není zde možná přílišná standardizace procesů. Uplatňují se zde vyčerpávající plánovací procesy, které jsou důležité pro roztřídění požadavků na jednotlivé produkty a vytvoření sekvencí ve výrobě. Důležitou charakteristikou zakázkové výroby je také to, že všechny zakázky neprocházejí stejným počtem operací a některé výrobky vyžadují různé operace (úkony) prováděné na jednom stroji. Každá úloha tedy může vyžadovat jiný sled operací (Lhoták, 2010). Schéma zakázkové výroby je uvedeno na obrázku (Obrázek 3).



Obrázek 3: Zakázková výroba-schéma

Zdroj: Gupta, 2014

Základní výhodou tohoto typu výroby je využívání jednoúčelových strojů, což umožňuje podniku vyrobit širokou škálu rozmanitých produktů (Lhoták, 2010). Díky náročnosti a rozmanitosti práce se zaměstnanci více rozvíjí a stávají se tak více kvalifikovanou a kvalitní pracovní silou. Jejich potenciál je zde tedy využit nadprůměrně. Další z výhod může být značný prostor pro využití kreativních metod ve výrobě a zavádění inovací. Naopak nevýhodou tohoto přístupu jsou značně vysoké náklady, které jsou spojené s rozmanitostí a flexibilitou výroby a také častým zaváděním změn. S tímto aspektem souvisí i vysoké skladové zásoby, které s sebou nesou další dodatečné náklady. Jak již bylo výše zmíněno, výroba zde vyžaduje náročné a detailní plánování, což považujeme také za určitou nevýhodu (Batch and Jobshop Production, 2014) (Planning And Control For Job Shop Production, 2014).

2.3 Optimalizace výroby

Optimalizací rozumíme proces, pomocí něhož se zkracuje a zjednodušuje již nalezená cesta k cíli. Základem je vytvářet a také používat rychlejší a jednodušší metody, operativnější vazby a struktury, s cílem zvýšení rychlosti a efektivnosti práce již zavedeného systému a tím docílit snížení nákladů organizace. V rámci optimalizace výroby je třeba se zaměřit na optimalizaci nejen procesů, ale také pracovišť. Optimalizací pracovišť, neboli linek rozumíme systematický proces snižování technologických procesů, odstraňování plýtvání a snížení výrobního času, což vede k růstu výkonu a produktivity práce. K optimalizaci pracovišť používáme různé nástroje jako jsou krátkodobé a dlouhodobé racionalizační opatření, které vybíráme na základě analýz a technik průmyslového inženýrství (Menlig a Koblasa, 2015).

Ke snížení výrobního času může sloužit například optimalizace layoutu neboli rozmístění strojů a například meziskladů ve výrobních halách. Minimalizují se tak přesuny mezi jednotlivými pracovišti, zkracuje se tak čas potřebný k výrobě a zároveň se snižují náklady výroby. V mé práci se však budu zaměřovat na optimalizaci samotného toku výrobního procesu. Budu se snažit seřadit zakázky ve výrobě tak, aby celkový čas jejich zhotovení byl minimální. Je to problém rozvrhování výroby a k jeho řešení můžeme používat různé algoritmy a přístupy. Nevhodné seřazení zakázek ve výrobě způsobuje velké prostoje a neefektivní využití výrobních kapacit podniku. Cílem je minimalizovat celkovou dobu vykonání všech zakázek (Pappová, 2010).

2.3.1 Přínosy optimalizace výroby

- minimalizace nákladů spojených s přestavbou strojů,
- identifikace kritických míst ve výrobě,
- klíčové indikátory výkonnosti (KPI) určují kritérium optimalizace,

- simulace různých scénářů / what-if analýza,
- snížení výrobních nákladů,
- snížení přesčasů,
- zkrácení průměrné doby realizace zakázky,
- snížení skladových zásob.

Výrobní systém musí vycházet z potřeb zákazníků zjištěných marketingovými prostředky. V současné době se prezentuje jako optimální tzv. logisticky řízená výroba neboli logisticky řízený podnik. Nejdůležitější úlohou logistiky ve výrobě je najít způsob, jak urychlit průchod materiálu výrobním procesem, a to s nejnižšími náklady. Pro výrobní logistiku je třeba znát průběh a délku jednotlivých částí výroby i celého výrobního cyklu (čas potřebný ke zhotovení výrobku, tedy čas od začátku první operace do skončení poslední). Sled jednotlivých cyklů za sebou poté nazýváme výrobním taktem (Pappová, 2010).

2.3.2 Výrobní náklady

Náklady výroby se dělí na fixní a variabilní. Fixními náklady jsou zejména odpisy výrobních zařízení. Tyto náklady jsou v logisticky řízeném podniku, oproti tradičně řízenému podniku, poměrně nízké. Z hlediska logistiky jsou zajímavé zejména náklady, které souvisí s variabilitou výrobní řady, což jsou náklady související s přestavováním strojů a jejich seřizováním. Variabilní náklady tvoří zejména mzdy pracovníků (seřizovačů), náklady na výměnné části zařízení a nástroje i náklady na potřebná mechanizační zařízení. Ve vyjádření na jednotku produkce pak mají tyto náklady v tradičně řízeném podniku strmější průběh než v podniku řízeném logisticky.

3 Rozvrhování výroby

V teorii rozvrhování výroby se vyskytují tyto tři základní pojmy (*stroj, operace a zakázka*)

Operace (*operation*) je základním technologickým úkonem, který již není možné dále dělit na částečné technologické úkony.

Zakázka (*job*) je posloupnost všech operací, které je třeba vykonat v rámci jedné zakázky.

Stroj (*machine*) je zařízení, které je schopné vykonávat jednu nebo několik operací najednou (Vaněčková, E., 1996).

3.1 Rozvrhování zakázkové výroby

Jedná se o problém přiřazení zakázek k různým strojům dle jejich charakteristiky a náročnosti. V tomto typu úlohy tedy vystupuje stroj M_j ($j=1, \dots, m$), dále zakázky J_i ($i=1, \dots, n$). Cílem úlohy je zpracovat všechny zadané zakázky, a to v co možná nejkratším celkovém čase.

K řešení tohoto problému je často využíván Ganttův diagram. Tento diagram může mít dvě podoby (strojově orientovaný nebo zakázkově orientovaný). Tato metoda je však spíše názorná a není tedy pro mou složitou optimalizaci ideální. Výsledné řešení však pro lepší prezentaci výsledků také zpracuji do Ganttova diagramu (Agrawal, Pattanaik a Kumar, 2012).

Každá jednotlivá zakázka reprezentuje zpracování určitého počtu kusů výrobku. Výrobní proces každého kusu se skládá z několika operací $O_{i1}, \dots, O_{in_i}$, kde n_i je počet operací. Jestliže výrobek prochází pouze jednou operací, pak J_i je reprezentována O_i a požadavky na zpracování p_i . Dále zde vstupuje datum vstupu zakázky do systému r_i , při kterém začíná první operace O_1 dané zakázky J_i . Každá operace O_{ij} může být provedena na stroji $\mu_{ij} \subseteq \{M_1, \dots, M_m\}$.

Všeobecně se optimalizace rozvrhovacích problémů uplatňuje na jednoúčelové nebo paralelní stroje. V našem případě se jedná vždy o tzv. paralelní stroje. Lisy využívané ve společnosti XY mají výměnné nástavce (Brucker, 1995). Tento fakt však činí optimalizaci složitější, protože téměř na každém lisu se dá dělat několik různých operací. Proto zde vstupuje další časový faktor d , který reprezentuje čas strávený na přenastavení lisu. Tento čas se budeme snažit co nejvíce zkrátit, protože nepřináší žádnou hodnotu do produktu. Můžeme ho tedy chápat jako plýtvání (Batch and Job-shop Production, 2014)

V neposlední řadě je zde definována nákladová funkce $f_i(t)$, pomocí které měříme náklady na zpracování zakázky J_i v čase t . Pro definování této funkce jsou ve většině případů využívány další faktory, jako datum dokončení zakázky d_i a váha w_i . Při optimalizaci musí být brány v úvahu všechny tyto faktory p_i , p_{ij} , r_i , d_i a w_i . Nesmí zde existovat dva časové intervaly překrývající se na jednom stroji nebo zde nesmí existovat duplicita zakázek přidělených k různým strojům. Poté, pokud výsledný rozvrh minimalizuje nebo naopak maximalizuje zadaná kritéria, je považován za optimální řešení problému (Majer, 2003).

Pro řešení problému budeme navrhovat algoritmus, který je specifikován pomocí tzv. třífázové klasifikace $\alpha|\beta|\gamma$, kde α reprezentuje strojové prostředí, β charakterizuje zakázku a γ udává optimalizační kritérium. Toto klasifikační schéma bylo představeno Grahamem (Graham a Lawler, 1979).

Každá zakázka J je charakterizována několika operacemi O , které je třeba zhotovit na strojích M . Tyto operace mají určitý sled, je zde tedy definováno, že například na operaci O_2 se může začít pracovat až po dokončení operace O_1 . Uvedeme si zde jednoduchý ilustrační případ pro lepší pochopení zadaného problému (Yamada, a Nakano, 1997).

V tomto případě zde máme 3 zakázky ($J=3$), které mají být zhotovené na 3 strojích ($M=3$). Každá z operací má určitý čas, potřebný k jejímu zhotovení, jak je možné vidět na obrázku (Obrázek 4).

Table 1: A 3×3 problem

job	Operations routing (processing time)		
1	1 (3)	2 (3)	3 (3)
2	1 (2)	3 (3)	2 (4)
3	2 (3)	1 (2)	3 (1)

Obrázek 4: Tabulka zadání problému 3x3

Zdroj: Yamada a Nakano, 1997

Ve výše uvedené tabulce je zapsána výroba 3 zakázek (J). Ke každé operaci (O) je přiřazen stroj (M), pro něj je zvoleno číselné označení bez závorek. V závorkách jsou pak uvedeny časy potřebné ke zhotovení určité operace. Tyto časy jsou zapsány v následující matici:

$$R = \begin{pmatrix} 3 & 2 & 3 \\ 2 & 3 & 4 \\ 2 & 1 & 3 \end{pmatrix}$$

Z uvedených informací a z matice jednoduše získáme v jakém pořadí prochází zakázka přes různé stroje. V tomto případě vypadá sled následovně:

$$J_1: M_1 \rightarrow M_2 \rightarrow M_3$$

$$J_2: M_1 \rightarrow M_3 \rightarrow M_2$$

$$J_3: M_2 \rightarrow M_1 \rightarrow M_3$$

Z tohoto zápisu je zřejmé, že například u zakázky číslo 1 je první operace přiřazena ke stroji číslo 4, druhá operace ke stroji číslo 3 a tak dále.

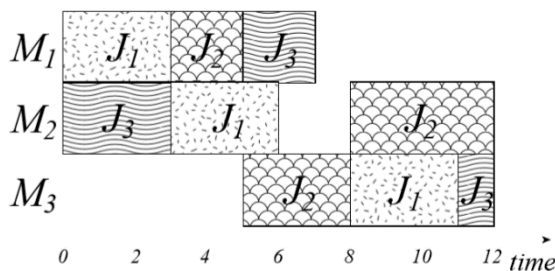
Podobným způsobem můžeme získat sled operací na jednotlivých strojích. V jakém pořadí se budou tyto operace zpracovávat.

$$M_1: J_1 \rightarrow J_2 \rightarrow J_3$$

$$M_2: J_3 \rightarrow J_1 \rightarrow J_2$$

$$M_3: J_2 \rightarrow J_1 \rightarrow J_3$$

Vzniklé řešení můžeme prezentovat i pomocí Ganttova diagramu, který je přehlednější a názornější, diagram je znázorněn na obrázku níže (Obrázek 5).

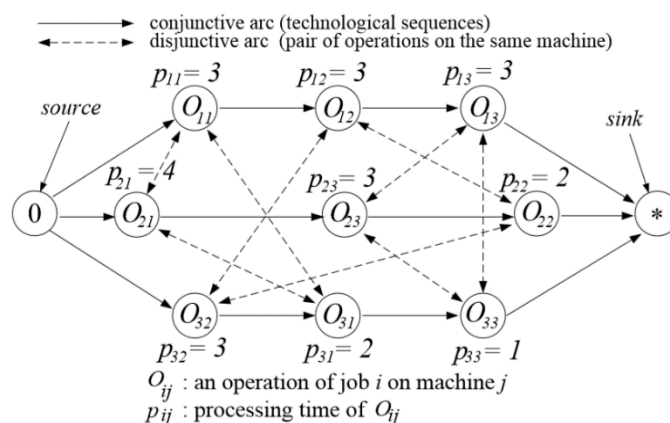


Obrázek 5: Ganttův diagram problému 3x3

Zdroj: Yamada a Nakano, 1997.

Disjunktivní graf se používá zejména k reprezentaci zadaného problému. Tento graf G je definován množinou uzlů V , dále množinou konjunktivních hran C a množinou disjunktivních hran D . V matematickém zápisu $G = (V, C \cup D)$. Uzly zde reprezentují všechny operace zadaných úloh (zakázek). Množina uzlů V obsahuje navíc dva speciální uzly, počáteční 0 a koncový uzel $*$. Tyto uzly jsou ohodnoceny nulami, protože samy nemají přiřazenou žádnou procesní dobu trvání operace. Ostatní uzly jsou tedy ohodnoceny dobami trvání jím odpovídajících činností (operací). Orientované konjunktivní hrany vyjadřují zadané pořadí operací v rámci jednotlivých úloh. Hrany vycházející z počátečního uzlu 0 , které směřují do uzlů odpovídajících prvním operacím zakázek a také hrany vycházející z posledních operací úloh, směřujícím do koncového uzlu $*$ (Haq, Balasubramanian, Sashidharan a Karthick, 2004).

Disjunktivní graf se také používá k vyřešení situace, kdy dvě operace mají být ve stejný čas zpracovávány na jednom stroji. Disjunktivní hrany zde spojují operace, které musejí být zhotoveny na stejném stroji. Na začátku algoritmu jsou disjunktivní hrany neorientované. Množina disjunktivních hran D tvoří m úplných podgrafů, z nichž každý náleží právě jednomu stroji (m zde znamená počet strojů) (Kwok a Ahmad, 1999). Disjunktivní graf k uvedenému případu je zobrazen na obrázku (Obrázek 6).



Obrázek 6: Disjunktivní graf

Zdroj: Yamada, T. and Nakano, R., 1997.

Pro lepší pochopení zápisu disjunktivním grafem zde uvádím podrobné dovysvětlení zápisu. Označení O_{11} znamená operace ze zakázky číslo 1, která je zpracována na stroji číslo 1. P_{11} vyjadřuje procesní dobu operace zakázky číslo 1, která probíhá na stroji číslo 1 (Bierwirth a Mattfeld, 1999).

Přerušované šipky potom znázorňují vztahy mezi operacemi, které se ve stejný čas potkají na stejném stroji. Při řešení takovéto úlohy musíme brát v potaz návaznost jednotlivých operací a sice, že 2. krok zakázky může být započat až po skončení operace 1. V tomto případě konkrétně 3.krok (O_3) zakázky 2 (J_2) na stroji 2 (M_2) musí počkat na dokončení předchozí operace (O_2), která se zpracovává na stroji 3 (M_3), proto zde vznikají dvě časové jednotky, kdy je výroba na stroji M_2 přerušena (Phanden, Jain a Verma, 2012).

Hlavní cíle rozvrhování výroby:

- dodržení termínů zhotovení zakázky určeného zákazníkem,
- minimalizovat zpoždění prací na zakázce,
- minimalizovat čas dokončení zakázky,

- minimalizovat dobu nečinnosti strojů (prostoje),
- minimalizovat držení zbytečných zásob,
- minimalizovat průměrnou dobu setrvání (průtoku) zakázky v systému,
- snížit náklady na přenastavení stroje.

3.2 Prioritní pravidla

V problematice rozvrhování výrobního procesu se používá několik druhů algoritmů. Tyto algoritmy se liší podle vstupu a výstupu jednotlivých zakázek do procesu výroby nebo jejich cestou samotným procesem výroby. Algoritmy existují buď přerušované, anebo nepřerušované. Nepřerušované algoritmy jsou takové kdy po vstupu zakázky do procesu výroby již nelze její zpracování přerušit, a to po určitý předem vyhrazený čas. Naproti tomu přerušované algoritmy jsou založeny na tom, že rozvrhovatel může výrobu zakázky s nižší prioritou přerušit kdykoliv, jestliže do systému vstoupí nová zakázka s vyšší prioritou, která je připravena na zpracování. Rozdělení algoritmů je uvedeno v následující tabulce (Tabulka 5).

Tabulka 5: Klasifikace algoritmů

Přerušované algoritmy	Nepřerušované algoritmy
First come first served (FCFS)	First come first served (FCFS)
Shortest job next (SJN)	Shortest job next (SJN)
Shortest remaining time	Priority based scheduling
Round Robin scheduling	

3.2.1 First come first served (FCFS)

Při použití tohoto algoritmu jsou zakázky odbavovány v tomtéž pořadí, v jakém vstoupily do systému. Děje se tomu tak bez jakýchkoliv jiných preferencí nebo zásahů do jejich pořadí. Zakázky, které čekají na vstup do procesu výroby, tvoří tzv. frontu. Tento systém je čteně využíván v běžném životě (v čekání na taxi, usazování hostů v restauraci, či u pokladen v obchodě).

Základní charakteristiky systému:

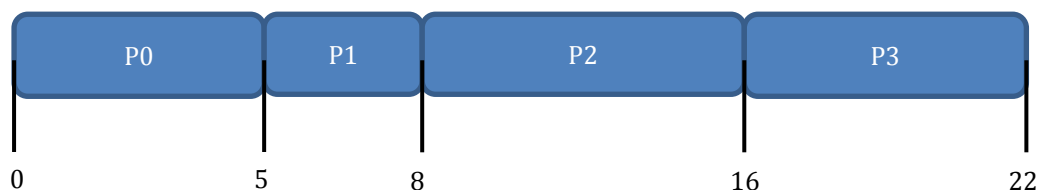
- zakázky jsou odbaveny stylem „první dovnitř, první ven“,
- jedná se o nepřerušovaný algoritmus,
- jednoduchý na pochopení a implementaci,
- jeho implementace je založená na bázi FIFO (first in, first out),
- nízká výkonnost (průměrná čekací doba je dlouhá).

Nákres metody First come, first serve, aplikované na jednoduchém příkladu, je uveden na obrázku (Obrázek 7), zadání tohoto příkladu je v tabulce (Tabulka 6) a řešení příkladu poté v Tabulka 7 (Operating System Scheduling algorithms, 2017).

Praktický příklad:

Tabulka 6: FCFS praktický příklad-zadání

Process	Arrival time	Execute time	Service time
P0	0	5	0
P1	1	3	5
P2	2	8	8
P3	3	6	16



Obrázek 7: FCFS nákres

Výpočet čekací doby:

Tabulka 7: FCFS praktický příklad-řešení

Process	Wait time: Service time-Arrival time
P0	0-0=0
P1	5-1=4
P2	8-2=6
P3	16-3=13

Průměrná čekací doba: $(0+4+6+13) / 4 = 5,75$

3.2.2 Shortest job next (SJN)

Tento algoritmu také můžeme znát pod názvem shortest job first.

- jedná se o nepřerušovaný algoritmus,
- nejlepší způsob, jak co nejvíce zkrátit dobu čekání,

- jednoduchý na implementaci v zakázkové výrobě, kde je požadovaná doba (CPU) předem známá,
- tento algoritmus není možno implementovat v interaktivních systémech, kde CPU čas není předem známý,
- plánovač musí vždy předem vědět kolik času proces zabere.

Podobný princip jako u shortest job first je u prioritního pravidla shortest remaining time. V podstatě jediný rozdíl mezi těmito pravidly je v tom, že u shortest remaining time algoritmu můžeme práci na objednávce přerušit, zatímco pravidlo shortest job next je nepřerušovaný algoritmus (Talbi, a Farouk, 2016). Nákres metody shortest job next uvádím na obrázku (Obrázek 8), zadání pro zhotovení nákresu je v Tabulka 8 a řešení s využitím metody SJN poté v Tabulka 9.

Tabulka 8: SJN praktický příklad-zadání

Process	Arrival time	Execute time	Service time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



Obrázek 8: SJN nákres

Výpočet čekací doby:

Tabulka9: SJN-praktický příklad řešení

Process	Wait time: Service time-Arrival time
P0	3-0=3
P1	0-0=0
P2	16-2=14

P3	8-3=5
-----------	-------

Průměrná čekací doba: $(3+0+14+5) / 4 = 5,50$

Nevýhodou tohoto algoritmu je, že někdy některá objednávka může v systému čekat věčně.

3.2.3 Priority based scheduling (PBS)

Jde o nepřerušovaný algoritmus a naroveň je to jeden z nejvíce používaných algoritmů v zakázkových systémech

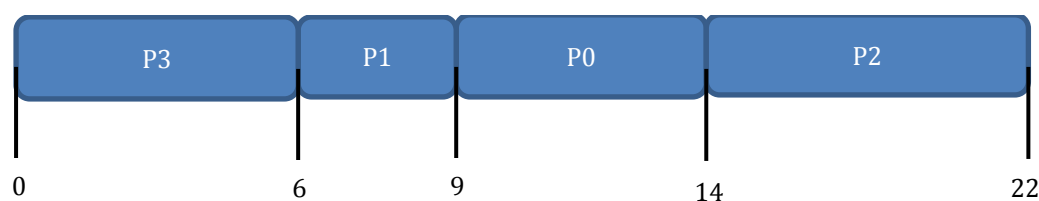
- Procesy jsou ohodnoceny váhou dle jejich priority, proces s nejvyšší prioritou je nutné odbavit jako první. Takto postupujeme u všech procesů v systému.
- Procesy se stejnou prioritou jsou dále seřazeny podle principu FIFO.
- Priority mohou být jednotlivým procesům přiřazeny na základě historických dat, časové náročnosti, atd (Operating System Scheduling algorithms, 2017).

Nákres principu metody Priority based scheduling je zobrazen na obrázku níže (Obrázek 9). Zdrojová data pro vytvoření nákresu jsou uvedena v Tabulka 10, výpočet času zhotovení zakázek je v tabulce (Tabulka 11).

Praktický příklad:

Tabulka 10: PBS praktický příklad-zadání

Process	Arrival time	Execute time	Priority	Service time
P0	0	5	1	9
P1	1	3	2	6
P2	2	8	1	14
P3	3	6	3	0



Obrázek 9: PBS nákres

Výpočet čekací doby:

Tabulka 11: PBS praktický příklad-řešení

Process	Wait time: Service time-Arrival time
P0	9-0=9
P1	6-1=5
P2	14-2=12
P3	0-0=0

Průměrná čekací doba: $(9+5+12+0) / 4 = 6,50$

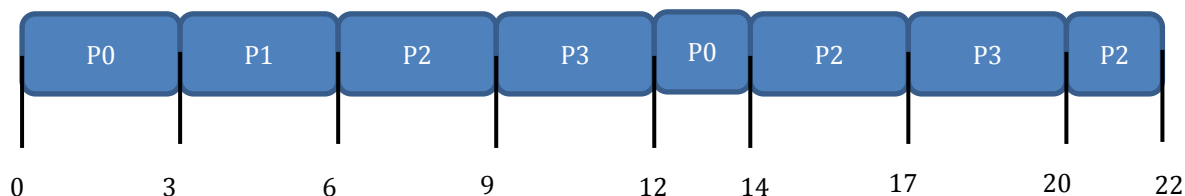
3.2.4 Round-Robin scheduling (RBS)

Jedná se o přerušovaný rozvrhovací algoritmus. Každý proces má pevně stanovenou dobu na zpracování, tuto dobu nazýváme quantum. Po započetí práce na procesu je proces po uplynutí určité doby (quantum) přerušen jiným procesem. Dochází tak k prolínání a střídání jednotlivých procesů (Talbi a Farouk, 2016), jak můžete vidět na obrázku. Jako zadání pro jednoduchý praktický příklad, zobrazený na schématu (Obrázek 10) slouží Tabulka 12 a výpočet je uveden v tabulce (Tabulka 13).

Praktický příklad:

Tabulka 12: RBS praktický příklad-zadání

Process	Arrival time	Execute time	Service time
P0	0	5	3
P1	1	3	0
P2	2	8	16
P3	3	6	8



Obrázek 10: RBS nákres

Výpočet čekací doby:

Tabulka 13: RBS praktický příklad-řešení

Process	Wait time: Service time-Arrival time
P0	$(0-0)+(12-3)=9$
P1	$3-1=2$
P2	$(6-2)+(14-9)+(20-17)=12$
P3	$(9-3)+(17-12)=11$

Průměrná čekací doba: $(9+2+12+11) / 4=8,50$

3.2.5 Porovnání přístupů

Podle výše uvedených výpočtů praktických příkladů usuzuji, že nejefektivnější je použití přístupu Shortest job next, neboli shortest job first. Při použití tohoto přístupu vyšla nejkratší doba čekání zakázek v pracovním procesu. U tohoto přístupu mají přednost zakázky s kratším časem zhotovení, jsou tedy vpuštěny do procesu výroby dříve, než ostatní zakázky s delšími časy. Druhý nejlepší výsledek byl zaznamenán u metody first come, first serve. Tato metoda je i v dnešní době stále velmi hojně využívána. Některé firmy ji používají automaticky, ačkoliv zde na příkladu vidíme, není vždy optimálním řešením. Naopak nejhoršího výsledku u stejného zadání příkladu dosáhla metoda Round-Robin based scheduling. Porovnání jednotlivých přístupů je uvedeno v následující tabulce (Tabulka 14):

Tabulka 14: Porovnání metod

Metoda	Průměrná čekací doba
First come, first serve	5,75h
Shortest job next	5,50h
Priority based scheduling	6,50h
Round-Robin based scheduling	8,50h

4 Metody řešení

Základem rozvrhovacích problémů je vždy požadavek na optimalizaci nějaké účelové funkce. Tato účelová funkce závisí na sekvencích jednotlivých operací na strojích. Jedná se o tzv. NP-těžké úlohy, k jejichž řešení je možné použít buď přesné metody (např. metodu větví a mezí) nebo heuristické metody (např. simulované žíhání, tabu search nebo genetické algoritmy). Přesné metody však přestávají být od určitého rozsahu problému efektivní, proto se ve většině případech přikláníme k použití metod heuristických (Talbi a Farouk, 2016).

4.1 Přesné metody

Historie použití přesných metod pro řešení rozvrhovacích problémů sahá až do roku 1954, kdy se Johnson zabýval rozvrhováním proudové výroby libovolného počtu úloh na dvou strojích. Tehdy vycházel z teoremu, že má-li permutace n úloh vlastnost, že platí $\min \{p_{i1}, p_{j1}\} \leq \min \{p_{i2}, p_{j2}\}$ pro $1 \leq i, j \leq n$, pak je pro tuto permutaci hodnota C_{max} optimální.

Za nejúspěšnější exaktní metodu pro řešení rozvrhovacích problémů je všeobecně uznávaná metoda větví a mezí, která je také označována jako metoda lineárního programování. Při této metodě se snažíme najít celkové nejlepší řešení X^* pomocí rozdělování původní množiny řešení X na menší podmnožiny a stanovením dolní meze hodnoty účelové funkce v každé podmnožině. Tyto podmnožiny jsou chápány jako množiny řešení podproblémů původního problému. Pokud je v některé z podmnožin stanovena dolní mez vyšší, než je dosud nalezené nejlepší řešení, můžeme tuto podmnožinu eliminovat. Obvykle se při této metodě používá jedna ze dvou prohledávacích strategií (prohledávání do šířky, prohledávání do hloubky) (Hart, 2008).

Další přesnou metodou, kterou je možné použít při řešení rozvrhovacích problémů je dynamické programování. U této metody je problém rozdělen na několik stupňů, kde v každém z těchto stupňů je třeba učinit rozhodnutí. Tato rozhodnutí mají posléze vliv na rozhodnutí, která budou učiněna v pozdějších stupních. Podle Bellmanova principu optimality je sestavena rekurzivní rovnice, která popisuje optimální hodnotu účelové funkce v daném stupni jako funkci hodnoty, získané v předchozím stupni (Jain a Meeran, 1998).

Použití přesných metod není vhodné u rozsáhlejších problémů, protože jejich časová náročnost roste exponenciálně a lineárním růstem rozsahu problému.

4.2 Heuristické metody

Pojem heuristika sahá až do dob Starého Řecka (z řečtiny *heuristikó*-nalézt, objevit). V dnešní době se název heuristika vykládá jako zkusmé řešení problémů, bez znalosti přesnějšího algoritmu nebo metody. Použitím heuristických metod sice nelze dosáhnout globálně optimálního řešení, ale v přiměřeném čase můžeme naleznout i několik suboptimálních řešení, jež pro nás mohou být také uspokojivé. Toto tvrzení však nelze dokázat, všeobecně heuristické metody používáme tam, kde neznáme lepší exaktní algoritmus pro získání přesného řešení (Hanzal, M., 2014). Heuristické metody pro řešení optimalizačních úloh se používají zejména pro optimalizaci mnohparametrových funkcí s mnoha extrémy nebo neznámým gradientem. Proces prohledávání prostoru řešení vyžaduje rovnováhu dvou cílů:

- Co nejrychleji najít nejbližší lokální optimum (minimum nebo maximum) v okolí výchozího bodu.
- Co nejlépe prohledat prostor všech řešení.

Příkladem využití heuristických postupů jsou úlohy jako například Problém obchodního cestujícího (*Traveling salesman problem*), ve kterém hledáme nejkratší možnou cestu, procházející všemi body na mapě (Šmerek a Moučka, 2008).

K posouzení kvality nalezeného řešení je třeba zavést tzv. hodnotící funkci, která je schopna ohodnotit všechna řešení z prostoru množiny možných řešení zadaného problému. Využití heuristických metod je tedy reálné pouze tam, kde je možné takovou funkci sestavit. Zde je příklad hodnotící funkce, vidíme zde, že funkce je zobrazením z množiny všech řešení X do množiny reálných čísel.

$$f(x): X \rightarrow R$$

Nejčastějšími heuristickými metodami, používanými pro řešení optimalizačních úloh jsou tzv. horolezecký algoritmus, zakázané prohledávání, simulované žíhání nebo genetické algoritmy (Jain a Meeran, 1998).

Přehled metod používaných k řešení rozvrhovacích problémů je uveden v následující tabulce (Tabulka 15):

Tabulka 15: Rozdělení heuristických metod

Deterministické metody	Stochastické metody	Smíšené metody
Horolezecký algoritmus	Simulované žíhání	Genetické algoritmy
Metoda větví a mezí	Zakázané prohledávání	

5 Metodika práce

Problematika řazení zakázek je velice rozsáhlá a komplikovaná. Je možné ji řešit několika exaktními technikami a metodami. Jednou z možností, jak nalézt optimální řazení zakázek pro daný způsob výroby může být například metoda Monte Carlo, využívající pseudonáhodná čísla. Tato metoda však vybírá vhodný scénář řazení zakázek do jisté míry náhodně. Tímto způsobem tedy nelze získat optimální řešení, pokud neprojdeme všechny možné scénáře řešení, výsledné řešení tedy nazýváme řešením suboptimálním.

Seřazení zakázek se v dnešní době běžně provádí za pomoci speciálních softwarů, které jsou navrženy a uzpůsobeny pro konkrétní problém v dané společnosti. Mezi tyto programy řadíme například software Mangrow Planner, který je založen na platformě Microsoft Planner. Tento systém je hojně využíván a v minulosti jej do výrobního procesu aplikovala například významná dodavatelská firma v automobilovém průmyslu, TRW (Merz, 2013).

Dalším informačním systémem využívaným pro plánování výroby a řazení zakázek je systém AROP, který využívá pro plánování a řízení výroby koncept MSO (modelování, simulace, optimalizace). Toto pokročilé dynamické plánování je výjimečně použitými metodami a snadným ovládáním ze strany uživatele.

Problematikou řízení výroby a řazení zakázek se zabývá několik specializovaných společností. Tyto společnosti klientům zpracovávají a aplikují informační systém a své řešení zadaného problému do prostředí organizace. Z těchto společností jmenuji například společnost Merica s.r.o., která se problematikou, nejen optimalizace výroby, zabývá již několik let a na českém trhu je jednou z nejvyhledávanějších (Merica, 2004).

Nejdříve budu přímo v organizaci ve výrobním prostředí pozorovat chod výrobních linek. Dále zde budu měřit a získávat důležitá charakteristická data o výrobních linkách, pracovnících, a také samotných zakázkách. Na tyto data poté použiji genetický algoritmus, který prokombinuje jednotlivé faktory zakázek určených do výroby. Algoritmus bude nastaven na minimalizaci celkového času zpracování všech zadaných zakázek. Tento algoritmus tedy vybere nejlepší možné seřazení zakázek s nejkratším celkovým časem. K aplikaci genetického algoritmu využiji programovací jazyk R.

Proběhne zde také porovnání účinnosti metody genetických algoritmů s jinou metodou vhodnou pro řazení zakázek ve výrobě. Konkrétně udělám porovnání s metodou lineárního programování, která na základě sestavení účelové funkce

a soustavy nerovnic omezujících podmínek generuje možné scénáře řešení, z nichž také nakonec vybírá optimální, nebo v dané situaci lépe řečeno suboptimální řešení. Výsledná řešení poté zakreslím do Ganttových diagramů, které prostřednictvím grafů názorně zachytí sled po sobě jdoucích činností na jednotlivých zakázkách.

Výsledkem mé práce by tedy mělo být jednak porovnání dvou metod pro řešení zadaného problému a doporučení pro zjednodušení postupu tvoření pracovních rozvrhů ve výrobních společnostech. Mým předpokladem je, že metoda genetických algoritmů bude pro problém řazení zakázek ve výrobě vhodnější, protože bere v úvahu více kritérií a metoda jako taková se mi zdá být propracovanější a přesnější.

6 Genetické algoritmy

Genetický algoritmus je heuristický postup, který je využitelný tam, kde neexistuje přesný postup pro nalezení exaktního řešení nebo kde by řešení úloh z praxe systematickým prozkoumáváním trvalo téměř nekonečně dlouho. Genetické algoritmy jsou stochastické optimalizační metody založené na darwinovském principu evoluce. Pod obecný pojem evoluční algoritmy se zařazují genetické algoritmy, evoluční strategie, evoluční programování a genetické programování, jako hlavní metodologie. U zrodu genetických algoritmů stála myšlenka, že při hledání lepších řešení složitých problémů by bylo možno obdobným způsobem kombinovat části již existujících řešení (Hynek, 2008).

6.1 Historie

Již v 19. století objevil genetické procesy Johann Gregor Mendel. Jeho myšlenky nadále rozvinul Charles Darwin, ve své knize *O vzniku druhů přirozeným výběrem čili zachování vhodných odrůd v boji o život*. Počítačová analogie genetických procesů, tedy první realizace genetických algoritmů se začaly objevovat v 70. letech 20. století. Vyvinutí počítačové verze genetických algoritmů je spojeno se jmény J. Hollanda a D. Goldberga. Teoretické aspekty využitelné v oblasti informatiky byly rozvedeny v knize *Adaptation in Natural and Artificial Systems* od Johna Hollanda, který se stal průkopníkem v oblasti genetických algoritmů. Rozšíření aplikace genetických algoritmů do oblastí řízení firem se objevuje během několika posledních let (Dostál, 2008).

6.2 Terminologie

Základním důležitým pojmem v oblasti genetických algoritmů je *Chromozóm (Genome)*, což je obecná genetická informace reprezentována sekvencí symbolů z nějaké abecedy (reálná čísla, znaky nebo jejich kombinace apod.), skládá se ze sekvenčně uspořádaných *Genů* (místo neboli pozice v chromozómu). Každý gen řídí dědičnost jednoho nebo několika znaků a jeho pozice v chromozómu má název *locus*. Chromozóm představuje tzv. genotyp a jeho význam, tj. informace zakódovaná v chromozómu, která název *fenotyp*. Každý nositel genetické informace se nazývá *Jedinec (Individual)*. Dále *Rodič (Parent)* je jedinec vybraný metodou selekce, následně vstupující do rekombinace a *Potomek (Offspring)* je jedinec, který vznikne na základě rekombinace dvou nebo více potomků. *Populace (Population)* je obecnější vyjádření pro skupinu jedinců. Důležité je zde *Schéma (Scheme)* jako předpis pro chromozóm-

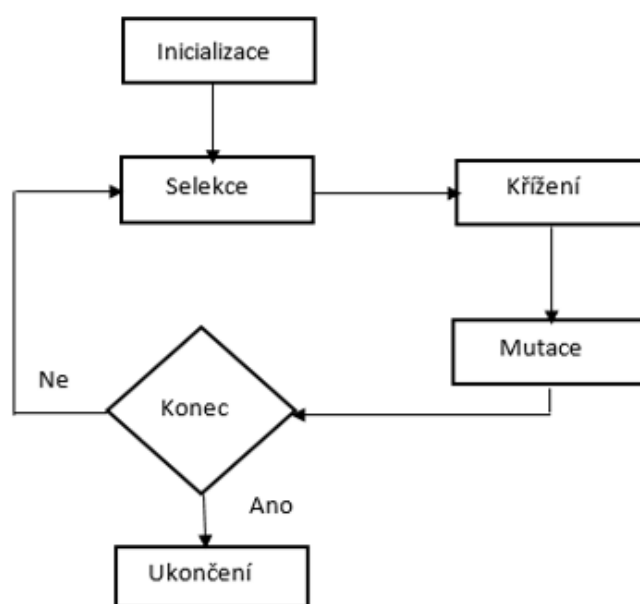
řetězec, v němž se alespoň na jedné pozici vyskytuje povolený symbol (v případě binární reprezentace je to 0 nebo 1) (Dostál, 2008).

Dalším důležitým termínem je tzv. *Ohodnocení (Fitness)*, tedy ohodnocení jedince, které rozhoduje o jeho přežití. Dále jsou zde genetické operátory, pomocí kterých prochází počáteční generace přeměnou. Mezi genetické operátory se řadí *Křížení (Crossover)*, kde z kombinace dvou nebo více jedinců, vznikne nový jedinec nebo jedinci, *Mutace (Mutation)*, což je náhodná změna alel nebo alely genů chromozómu a *Selekce (Selection)*, neboli výběr chromozómů, které se stanou rodiči (pro další generaci nebo rekombinaci) (Mach, 2009).

6.3 Základní funkce a principy

Princip metody Genetických algoritmů spočívá v tom, že se nejprve vytvoří počáteční populace m chromozómů. Poté se tato populace mění pomocí genetických operátorů tak dlouho, dokud není proces ukončen, např. počtem cyklů (generací).

- Vytvoření nulté generace-inicializace (obvykle náhodně vygenerována, lze zaznamenat i prvopočáteční řešení, ne vždy však musí řešení konvergovat).
- Výběr vhodných jedinců z populace-selekce (operace výběru), volí se takový jedinec, již jsou relevantní k dalšímu řešení.



Obrázek 11: Základní princip genetických algoritmů

Vlastní zpracování, zdroj: Dostál, 2008

- Generování nových jedinců-křížení, mutace.
- Omezení počtu jedinců na velikost nastavené populace.
- Vypočtení zdatnosti jedinců-fitness.
- Ukončení cyklu-konec, pokud je splněna zastavovací podmínka, v opačném případě pokračuj bodem 2 (Hanzal, 2014).

Při ukončení algoritmu se vybere jedinec s tzv. nejvyšší zdatností, ten pak reprezentuje nejlepší nalezené řešení. Vlastní průběh algoritmu ve zjednodušené podobě popisuje výše uvedené schéma (Obrázek 11).

6.3.1 Vytvoření počáteční generace

Způsob kódování jedince hraje významnou roli při úspěchu nebo neúspěchu Genetického algoritmu v aplikaci na konkrétní úlohu. Nejpoužívanější způsob kódování je binární kódování, jeho výhodou je zejména jednoduchost a přirozená reprezentace na počítači. Takto vyjádřený chromozóm je složený z genů, které nabývají hodnot 0 nebo 1 (Tabulka 16) reprezentuje zakódování populace se dvěma jedinci (Škrabal, 2010).

Tabulka 16: Jednoduché zakódování populace

Jedinec číslo	Chromozóm jedince
1	(0, 1, 0, 0, 0, 1, 0, 1, 1, 1)
2	(1, 0, 1, 1, 1, 0, 0, 1, 0, 0)

- Fitness funkce

Ohodnocením jedince máme na mysli výpočet jeho kvality neboli jeho dekadické vyčíslení. Hodnocení jedinců je možné udělat na základě účelové funkce, ale i mnoha jinými způsoby. Pokud máme například dostatečně velkou populaci, je možné ji rozdělit na dvě a jedince vzájemně porovnat. Obvykle je to reálné číslo v rozsahu od 0 do 1 (Hynek, 2008). Příklad ohodnocení fitness funkce je uveden v tabulce níže (Tabulka 19).

Tabulka 17: Ohodnocení funkcí fitness

Jedinec číslo	Chromozóm jedince	Ohodnocení $f(i)$	% z celkového ohodnocení $p(i)$	Kumulované ohodnocení
1	(1,1,0,0,1,1,0,1,1,1)	7	41,17 %	0,4117
2	(1,1,1,1,0,1,0,0,0,0)	3	17,64 %	0,5881

3	(1,1,1,0,0,0,0,1,0,1)	6	3,29 %	0,9410
4	(0,0,1,1,0,0,1,1,0,0)	1	5,88 %	1,0000

6.3.2 Výběrové metody (selekce)

Pomocí operátoru selekce jsou vybíráni jedinci, kteří se mají rozmnožovat a tvořit tak další generaci. Metody selekce by měly respektovat pravidla přirozeného výběru, dle Darwinovy teorie. To znamená, že do procesu reprodukce jsou vybíráni silnější jedinci, protože mají větší šanci se prosadit v konkurenci, a ti se reprodukují na úkor těch slabších. Avšak přílišné upřednostňování například silnějších jedinců, by mohlo znamenat uváznutí v lokálním minimu, naopak přílišná rozmanitost (diversita) by nezaručovala rychlou konvergenci k optimu. Důležité zde tedy je, že se nejedná o jednoduché vybírání těch nejlepších jedinců z populace, ale selekce musí zaručit i to, že se rekombinace bude moci zúčastnit i ten nejhorší jedinec z populace (Dorndorf a Pesch, 1995). Metod, jak provést selekci je několik, některé z nich popíši zde:

- Ruletový výběr (Roulette wheel selection)

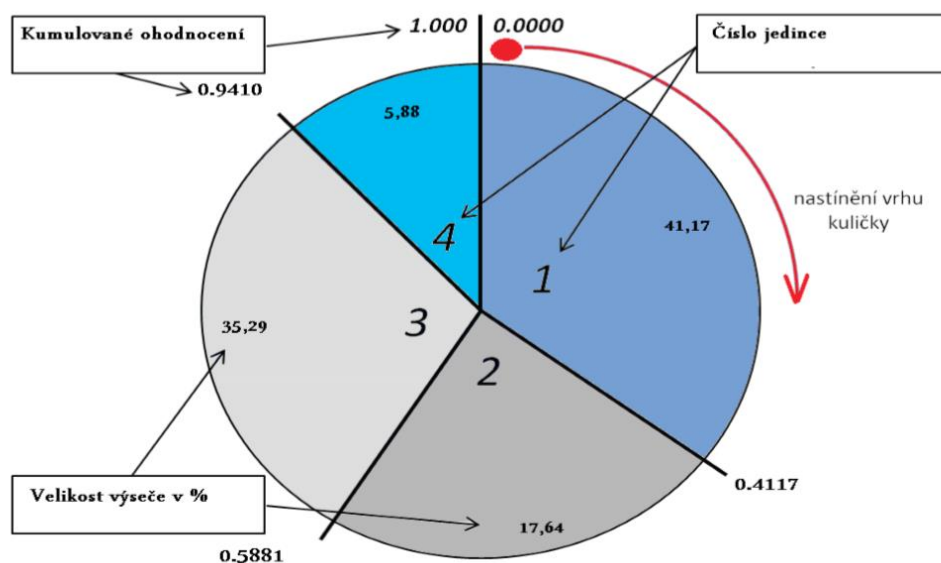
Výběr jedince zde závisí na pravděpodobnosti vylosování řetězce při tak zvané vážené ruletě. Obvod rulety je zde rozdělen do oblastí různých velikostí podle hodnocení jedince (fitness). Pravděpodobnost přežití jedince je tedy přímo úměrná jeho kvalitě. Na obrázku (Obrázek 12) je při výběru jedinců aplikovaný náhodný výběr jedince. Pokud je jedinec vybrán, padne na něj kulička ruletového kola a postupuje do další generace. Pravděpodobnost vylosování daného řetězce zjistíme tak, že nejdříve sečteme všechny fitness hodnoty a poté tímto součtem podělíme fitness hodnotu pro jednotlivé jedince (Dostál, 2008).

$$p(i) = \frac{f(i)}{\sum_1^N f(j)}$$

$P(i)$ pravděpodobnost výběru i -tého jedince

$f(i)$ fitness hodnota i -tého jedince

Tímto způsobem jsou postupně vybráni noví jedinci pro tvorbu nové generace, v závislosti na tom, do kterého intervalu nám kulička padne.



Obrázek 12: Ruletový výběr

Zdroj: Škrabal, 2010

Tento selekční mechanismus favorizuje nejsilnější jedince a u některých typů konkrétních úloh může uváznout rychle v lokálním optimu. Další nevýhodou tohoto selekčního druhu je, že zvládá pouze maximalizační úlohy, má problém se škálováním, a také způsobuje velké vzorkovací chyby a pro správnou funkci vyžaduje relativně velké populace (Dorndorf a Pesch, 1995). Z těchto důvodů byly vyvinuty i další selekční schémata.

Dostí podobnou metodou výběru je tzv. Stochastický univerzální výběr. Při tomto výběru není, na rozdíl od ruletového výběru, generován stejný (jako počet jedinců) počet náhodných čísel, ale je zde vygenerováno jen jedno náhodné číslo, které je následně dělené počtem jedinců. Toto náhodné číslo se nachází v intervalu $<0,1>$. Výsledné číslo určuje prvního vybraného jedince, ostatní jsou vybráni v po sobě jdoucích intervalech o konstantní délce $1/N$ ($N = \text{počet jedinců, které chceme vybrat}$). Prostřednictvím tohoto výběru je částečně zajištěna rozmanitost výběru, to znamená, že zde nejsou tolik favorizováni jedinci s nejlepším ohodnocením (Abu-Srhnah a Al-Hasan, 2015).

- Turnajový výběr (Tournament selection)

Jedná se o nejpoužívanější selekční metodu při praktické aplikaci genetických algoritmů. Její hlavní výhodou je jednoduchost implementace při zachování podmínek

selekčního mechanismu. Z populace se náhodně vyberou převážně dvojčlenné skupinky jedinců a ti jsou podrobeni souboji, jedinec s vyšší hodnotou fitness přežívá a postupuje do dalšího kola, názorně je vidět v tabulce (Tabulka 20). Na základě této skutečnosti je splněn předpoklad, aby přežívali lépe ohodnocení jedinci, ale zároveň je splněna diversita, protože jedinci jsou k souboji vybírání náhodně (Dostál, Reis a Sojka, 2005). Do turnajového výběru můžeme také zakomponovat náhodu, tedy vítězný jedinec postoupí nebo ne, bez ohledu na jeho fitness hodnocení, jak je vidět níže (Škrabal, 2010).

Tabulka 18: Turnajový výběr

Jedinec číslo	Ohodnocení jedince	Jedinci vybraní do souboje	Výherce souboje	Poznámky
1	5	(1,8)	8	
2	11	(3,4)	4	Jedinec č. 3 měl smůlu
3	14	(3,9)	3	
4	4	(4,9)	4	
5	2	(1,2)	2	
6	16	(2,3)	3	
7	8	(9,8)	8	7 již jednou postoupil
8	8	(5,7)	7	Náhodný výběr
9	1	(2,5)	5	Jedinec č. 2 měl smůlu

- Pořadová selekce

Zde jsou jedinci nejdříve seřazeni vzestupně, dle hodnoty jejich fitness funkce, přičemž poslední n -tý jedinec je ohodnocen hodnotou $H > 0$. Jedinci jsou poté vybírání na základě jejich pozice v realizovaném seřazení. Výhodou této metody selekce je omezení vlivu nadprůměrných jedinců, ovšem možnou nevýhodou může být fakt, že mapováním jedinců pouze na základě jejich pozice v řadě se ztrácí informace o skutečné kvalitě jedince (Dostál, Reis a Sojka, 2005).

- Další metody

Elitismus-tato metoda slouží k zachování nejsilnějších jedinců, aby se předešlo jejich vyřazení. Jedinci s nejlepším hodnocením fitness tedy postupují automaticky do nové generace.

Ořezávání–zde jsou jedinci dle svého ohodnocení seřazeni a následně rozděleni do 2 či více skupin. Do reprodukce poté postupují pouze jedinci z nejúspěšnější skupiny.

Náhodný výběr–v rámci této metody jsou jedinci k postupu do další generace vybíráni zcela náhodně (Hynek, 2008).

6.3.3 Operátory křížení

Pomocí operátoru křížení jsou z vybraných jedinců generováni potomci stávající generace. Při křížení dochází k výměně informací mezi dvěma řetězci. Proces křížení probíhá tak, že jsou nejdříve vybrány dva náhodné řetězce, které vstupují do rekombinace. Následně jsou prokombinovány části těchto jedinců tak, že vznikají dva úplně noví unikátní jedinci (Dorndorf a Pesch, 1995). Důležité zde také je, že toto křížení jedinců by mělo respektovat kvalitu jedince podle selekčních metod popsanych výše. Tímto způsobem tedy vznikají dosud nevytvořené kombinace řešení. Křížení je možno provést hned několika způsoby.

- Jednobodové křížení

Z předchozí etapy selekce máme vybrány dva chromozomy. Náhodně zvolíme bod v řetězci, ten tvoří hranici a rozdělí tak chromozom na dvě části. Tyto části se následně mezi potomky vymění. Názorně je tento proces popsán v Tabulka 19 (Dostál, 2008).

Tabulka 19: Jednobodové křížení

Původní chromozómy (rodiče)		Nové chromozómy (potomci)
(1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0)	→	(1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1)
(1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1)	→	(1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0)

- Dvou a vícebodové křížení

Dvoubodové křížení je podobné jako jednobodové. Rozdíl je v tom, že řetězec chromozomu je zde rozdělen pomocí dvou nebo více hranic na tři nebo více částí. V případě dvoubodového křížení pak od druhého bodu nedochází k záměně genů. Toto křížení je možné názorně vidět v Tabulka 20 (Kvasnička, Pospíchal a Tiňo, 2000).

Tabulka 20: Dvoubodové křížení

Původní chromozómy (rodiče)		Nové chromozómy (potomci)
(1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0)	→	(1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0)
(1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1)	→	(1, 0, 0, 0, 1, 0, 1, 0, 1, 0, 1)

Podobně dochází k vícebodovému křížení. Každý lichý bod zde určuje pozici, od které jsou geny chromozómů prohazovány a každý sudý bod křížení pak bod, kde dojde k přerušení záměny genů. Tento proces lze sledovat níže (Tabulka 21).

Tabulka 21: Vícebodové křížení

Původní chromozómy (rodiče)		Nové chromozómy (potomci)
(1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 0)	→	(1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0)
(1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1)	→	(1, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1)

- Uniformní křížení

Tento způsob křížení používá tak zvanou křížící masku, která má stejný počet znaků v řetězci jako vstupující chromozomy a má také podobu binárního kódu. Každé místo v řetězci je potenciálním bodem křížení. Jsou zde dvě odlišné křížící masky, tedy každý z rodičů má vlastní křížící masku. Platí zde pravidlo, že druhý rodič má vždy inverzní křížící masku než první rodič. Způsob konstrukce nového potomka je popsán v Tabulka 22. Každý gen, který potomek získá, je zde náhodně vybrán s určitou pravděpodobností (Goncalves, Mendes a Resende, 2002).

Tabulka 22: Uniformní křížení

Původní chromozómy (rodičovské)	Křížící maska	Nové chromozómy (potomci)
(1, 1, 0, 0, 1, 1, 1, 1, 1, 1)	(1, 0, 1, 0, 1, 1, 1, 1, 1, 0)	(1, 0 0, 0, 1, 1, 1, 1, 1, 1)
(0, 1, 0, 0, 1, 0, 1, 1, 0, 1)	(0, 1, 0, 1, 0, 0, 0, 0, 0, 1)	(0, 1, 0, 0, 1, 0, 1, 1, 0, 1)

Existuje zde mnoho dalších metod křížení, tyto metody jsou složitější a přesahují tak rámec mé práce. Nebudu je zde tedy blíže popisovat.

- křížení s částečným přiřazením,
- křížení s rekombinací hran,
- křížení se střídáním pozic,
- křížení se zachováním prostředí,

- cyklický operátor křížení.

6.3.4 Operátory mutace

Na výslednou novou generaci se na závěr aplikuje poslední operátor, mutace. Je možné, že za pomoci operátorů selekce a křížení dosáhneme velmi kvalitního řešení, ne však vždy se tomu tak stane. Tato situace může nastat například v případě, kdy máme stanovenou splňující podmínku, která pro ukončení procesu vyžaduje, aby na určitém místě v řetězci chromozomu byla např. 1, ale všechna vytvořená řešení zde mají 0. Z tohoto důvodu se poté aplikuje právě operátor mutace, kdy jednoduchou záměnou tohoto genu dosáhneme splňující podmínky (Dostál, Reis, and Sojka, 2005). Pomocí mutace se do nově vzniklých chromozómů vnáší prvek náhodné změny. Mutace napomáhá tomu, aby výsledné řešení neuvízlo v lokálním extrému. Na základě tohoto operátoru se můžeme generovat řešení, která nejsou na základě křížení dosažitelná. Mutace bývá většinou jednobodová, kdy v řetězci chromozomu je provedena náhodná změna jednoho genu (Minařík, 2007). Opět existuje několik druhů mutace:

- Binární mutace

Zde dochází v případě binárního kódování ke změně genů v řetězci z 1 na 0 a naopak. Tabulka 23 reprezentuje 5 % možnost změny u každého genu.

Tabulka 23: Binární mutace

Chromozóm před mutací:	(1, 1, 0, 0, 1, 0, 1, 1, 1, 1)
Chromozóm po mutaci:	(1, 1, 0, 0, 1, 0, 1, 0, 1, 1)

Počet mutací v chromozomu je možné zvyšovat při stagnující populaci a při následném zlepšení opět vrátit na původní hodnotu. Tato metoda mutace je v praxi nejpoužívanější.

- Výměnná mutace

Zde se jedná o záměnu dvou genů v řetězci. Tato metoda je dobře použitelná u tzv. permutačního kódování.

- Posuvná mutace

V řetězci je náhodně zvolen gen, a ten je posunut na náhodně vygenerovanou pozici, tímto úkonem se posunou pozice genů. Tento druh mutace je vhodný pro všechny typy kódování.

- Inverzní mutace

U inverzního typu mutace jsou nejdříve zvoleny náhodně dva geny v řetězci a následně dojde k invertování úseku mezi těmito geny (Hynek, 2008).

6.3.5 Reprodukce

Poslední operace v rámci tvorby nové generace. V této fázi vybíráme řetězce, jejichž fitness ohodnocení je nejvyšší. Tyto chromozomy jsou poté vystaveny opětovné selekci, křížení a mutaci. Naopak chromozomy s nízkou hodnotou fitness, které by nebyly přínosem pro další práci, jednoduše nahradíme kopiemi silnějších jedinců.

6.4 Praktické využití GA

V praxi se genetické algoritmy využívají k řešení ekonomických a technických úloh. Z ekonomického oboru lze uvést jako příklad výběr investic, sestavení investičního portfolia, které je složeno z více investic s různou výnosností a rizikem. Dalším příkladem je problém umístění distribučního skladu, kde je cílem umístit distribuční sklad tak, aby bylo zvolené místo optimální vzhledem k propustnosti distribučních cest a velikosti závazků (Dostál, Reis a Sojka, 2005). Genetické algoritmy se také využívají u tzv. úloh o baťožu, kde je cílem umístit výrobky o předem daných rozměrech do vymezeného prostoru. S takovými úlohami se můžeme setkat u distribučních zásilkových služeb například při volbě vhodného obalového materiálu. Asi nejznámějším a nejčastějším využitím genetických algoritmů je řešení problému obchodního cestujícího, kde hledáme optimální, co možná nejkratší trasu pro zavezení zadaných míst. Podobnou úlohou je hledání spojnice optimální trasy mezi dvěma vzdálenými městy. Předmětem mé práce je optimalizace výrobního plánu. Pro tento typ úlohy jsou genetické algoritmy čteně využívány. Ve výrobních společnostech se tyto algoritmy používají například pro návrh konstrukčního kusovníku, návrh řezného plánu, což je tak zvaná plošná optimalizace, a také pro plán tavení. V technické oblasti se genetické algoritmy využívají pro optimalizaci manipulačního procesu nebo optimalizaci fuzzy systémů. Zajímavé je také jejich využití při návrhu energeticky úsporného domu (Chryssolouris a Subramaniam, 2001).

7 Lineární programování

Lineární programování se řadí mezi metody operačního výzkumu, které se zabývají řešením rozhodovacích problémů, v nichž jde o určení intenzit realizace procesů, které probíhají nebo mohou probíhat v daném systému. Realizaci těchto procesů ovlivňují četné omezující podmínky, které je nutno respektovat. Cílem je nalézt takové řešení, kdy je cíl rozhodování splněn co nejlépe a zároveň jsou splněny i všechny omezující podmínky. Název lineární programování v podstatě vyjadřuje jakési plánování či přesněji vytváření scénářů budoucího vývoje, přičemž vazby (matematické funkce) v použitých modelech jsou vazbami lineárními. Tato metoda tedy slouží jako prostředek pro plánování realizace určitých procesů nebo činností, které zabezpečují dosažení optimálního výsledku ve vztahu k definovanému cíli.

Typickým využitím lineárního programování je optimalizace výrobního plánu firmy, směšovací úlohy, určení strategie reklamy nebo třeba optimalizace distribuce zboží (Lagová a Jablonský, 1999).

Metoda lineárního programování je rozdělena na dvě hlavní části, a sice definici tzv. ekonomického modelu, který je spíše samotným zadáním řešené úlohy a následně vytvoření matematického modelu, kde již pracujeme s konkrétními zadanými údaji a tvoříme matematické rovnice. Přehled základních informací ohledně těchto modelů je uveden níže v tabulce (Tabulka 24).

Tabulka 24: Prvky ekonomického a matematického modelu

	Ekonomický model	Matematický model	Příklady
Čeho chceme dosáhnout?	cíl analýzy	účelová funkce	maximalizace zisku minimalizace nákladů minimalizace rizika
Co můžeme ovlivnit?	procesy	proměnné	velikost produkce přepravované množství velikost investice
Jaké jsou překážky?	činitelé	omezující podmínky	daná výrobní technologie omezená zásoba surovin kapacita lidských zdrojů finanční rozpočet

Vlastní zpracování, zdroj: Lineární programování-grafické řešení

7.1 Fáze aplikace lineárního programování

Při aplikaci lineárního programování na řešení reálného rozhodovacího problému rozlišujeme několik základních, na sebe navazujících fází, které jsou také zobrazeny na obrázku (Obrázek 13):

Rozpoznání problému – Prvním podstatným krokem aplikace modelů operačního výzkumu je poznání problému v rámci reálného systému a jeho definice. V této fázi je třeba soustředit se na role vedoucích pracovníků na různých úrovních, kteří jsou schopni rozpoznat problém a odhadnout potřebu modelového přístupu pro analýzu tohoto problému. V kompetenci těchto pracovníků je také tvoření týmů příslušných odborníků, kteří se na této analýze budou podílet (Ferguson, 2015).

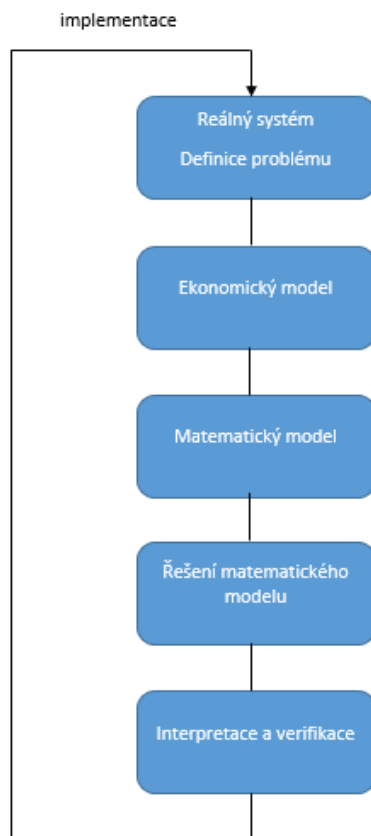
Formulace ekonomického modelu – Reálná problém je ve většině případů příliš složitý, není tedy úplně možné postihnout všechny jeho stránky. Ekonomický model poskytuje tedy jakési zjednodušené vyobrazení zadaného problému.

Formulace matematického modelu – V této fázi je třeba zadání problému převést z ekonomického modelu na model matematický, a tak ho formalizovat. Matematický model se dá následně řešit standartními postupy.

Řešení matematického modelu – Matematický model řešíme různými postupy a metodami operačního výzkumu, které jsou v dnešní době zabezpečeny kvalitními programovými systémy. Uživatelé mají v tomto kroku pouze vedlejší roli, kdy vybírají vhodný programový prostředek, který bude použit pro řešení modelů a jejich formální obsluhu.

Interpretace a verifikace výsledků – Právě interpretace získaných výsledků je mnohdy složitější než samotný výpočet odpovídajícího řešení. Verifikací poté ověříme, zda byl ekonomický a matematický model sestaven správně.

Implementace výsledků – Výsledné řešení nakonec implementujeme v rámci analyzovaného reálného systému. Úspěšná implementace by měla přispět ke zlepšení fungování daného systému s ohledem na sledovaný a definovaný cíl (Jablonský, 2007).



Obrázek 13: Fáze při aplikaci operačního výzkumu

Vlastní zpracování, zdroj: Jablonský, 2007.

7.2 Ekonomický model LP

Ekonomický model je kvantitativní a kvalitativní popis situace, který je výsledkem rozboru ekonomické reality. Dá se říci, že se jedná o vlastní slovní zadání úlohy. Ekonomický model by měl obsahovat cíl, procesy a činitele. Cíl neboli kritérium zadaného problému musí být jasně vymezen, příkladem může být maximalizace zisku nebo minimalizace nákladů. Dále zde musí být popsány procesy, které probíhají v dané rozhodovací situaci. Problémy sledujeme na zjednodušených modelech, které zjednodušují reálné problémy tak, že uvažuje pouze ty stránky skutečnosti, které jsou z hlediska zkoumaného jevu významné. Nakonec zde musí být zadány činitelé,

které ovlivňují daná proces, a také popis vztahů mezi procesy a činiteli (Hliněný, 2007).

7.2.1 Vzorový příklad-ekonomický model

Pro lepší vysvětlení formy ekonomického modelu zde uvádím zadání vzorového příkladu.

Firma balící bonboniéry má k dispozici 60 čokoládových, 60 oříškových a 85 karamelových bonbónů. Může vyrábět dva druhy bonboniér. Do první bonboniéry se dávají dva čokoládové, šest oříškových a deset karamelových bonbónů a do druhé deset čokoládových, šest oříškových a pět karamelových. Firma má vykalkulováno, že na každém kusu první bonboniéry vydělá 30 Kč a na druhé 45 Kč. Jaký je optimální výrobní program firmy, pokud chce maximalizovat zisk (Klicnarová, 2006)?

7.3 Matematický model LP

Matematický model se skládá z účelové funkce a omezujících podmínek, přičemž obě tyto části jsou vyjádřeny lineárními vztahy s konstantními koeficienty u jednotlivých proměnných současně s pravými stranami soustavy omezení.

Jedním ze základních pojmů v oblasti lineárního programování je tzv. vektor řešení. Vektor řešení označuje řešení optimalizačního problému. U problému určování optimální výrobní struktury firmy vyjadřují složky vektoru například množství jednotlivých druhů výrobků.

Příklad vektoru řešení:

$$\mathbf{x} = (x_1, x_2, x_3, \dots, x_n)$$

Za optimální je považována taková struktura výroby, při které rozhodující veličina nabývá extrémní hodnoty. Uvažujeme-li jako rozhodující veličinu zisk, poté je optimální struktura s maximální dosaženou hodnotou zisku. Pokud uvažujeme jako rozhodující veličinu náklady, optimální struktura je taková, která s sebou přináší minimální náklady. Rozhodující veličinou může být například i čas. Tuto rozhodující veličinu vyjadřujeme pomocí tzv. účelové funkce. Účelová funkce je funkce vektoru řešení, u výše uvedeného příkladu hledání optimální výrobní struktury je účelová funkce definována jako závislost výše zisku na množství jednotlivých výrobků v portfoliu (Friebešlová, 2006).

Příklad účelové funkce:

$$z = f(\mathbf{x})$$

Extrém účelové funkce (minimum nebo maximum) hledáme za určitých omezujících podmínek, které mají vliv na velikost složek vektoru řešení. Rozlišujeme dva druhy

omezujících podmínek, a sice vlastní omezující podmínky a omezující podmínky nezápornosti. Kdy vlastní omezující podmínky mohou být například limitovaná množství surovin pro výrobu určitého výrobku.

Příklad vlastní omezující podmínky:

$$h_i(\mathbf{x}) \leq a_i, h_i(\mathbf{x}) \geq a_i, h_i(\mathbf{x}) = a_i$$

kde $h_i(\mathbf{x})$ je funkce, která vyjadřuje např. závislost spotřeby určitého zdroje na vyrobeném množství daného výrobku a a_i jsou konstanty představující například disponibilní množství surovin. Dále jsou zde omezující podmínky nezápornosti, které platí pro všechny složky vektoru řešení.

7.3.1 Vzorový příklad-matematický model

Zde pokračuji v řešení zadaného příkladu z předchozí kapitoly. Hledaná proměnná x_1 zde zastupuje bonboniéru 1. typu a proměnná x_2 tedy bonboniéru 2. typu. Zároveň ze zadání (ekonomického modelu) víme, že zisk z prodeje první bonboniéry je 30Kč a prodej jednoho kusu druhé bonboniéry nám přinese zisk ve výši 45Kč.

Základem je určení účelové funkce modelu tak, že hledáme takovou kombinaci bonboniér 1. a 2. typu, která nám přinese největší zisk. Matematický model definuje účelovou funkci, která vyplývá ze zadání jako:

$$30x_1 + 45x_2 \rightarrow \max$$

Nyní definujeme vlastní omezující podmínky modelu jako:

$$2x_1 + 10x_2 \leq 60$$

Tato podmínka upravuje stav čokoládových bonbónů, konkrétně říká, že do bonboniéry 1. typu jsou třeba 2 čokoládové bonbóny, zatímco do druhé bonboniéry patří čokoládových bonbónů 10 kusů. Celkem máme k dispozici 60 kusů čokoládových bonbónů. Tedy součet kusů bonboniér vynásobený počtem čokoládových bonbónů obsažených v každé z nich musí být menší nebo roven 60 kusům.

$$6x_1 + 6x_2 \leq 60$$

Podobně jako u předchozí podmínky definujeme i podmínku týkající se oříškových bonbónů. Do každého typu bonboniéry patří přesně 6 kusů oříškových bonbónů. Opět můžeme sestavit bonboniéru z maximálního počtu 60 kusů oříškových bonbónů.

$$10x_1 + 5x_2 \leq 85$$

Poslední vlastní podmínka se týká karamelových bonbónů. Do první bonboniéry patří 10 kusů těchto bonbónů a do bonboniéry druhého typu jich patří pouze 5. Podmínka omezuje počet kusů karamelových bonbónů na 85 (Klicnarová, 2006).

Nakonec je třeba vždy definovat podmínku nezápornosti výsledných řešení.

$$x_1, x_2 \geq 0, \text{ celočíselné}$$

Vektor řešení:

$x = (x_1, x_2, \dots, x_n)^T$ – optimální řešení optimalizačního problému. Jednotlivé složky určují např. optimální množství výroby jednotlivých druhů výrobků.

Cenový vektor:

$c = (c_1, c_2, \dots, c_n)^T$ – většinou vektor jednotkových zisků či nákladů. V našem případě $(30, 45)^T$

Účelová funkce:

$z = f(x) = c^T x$ – funkce vektoru řešení. Vyjadřuje např. zisk v závislosti na objemu výroby, náklady, množství odpadu apod.

Poznámka k modelu:

Na účelovou funkci mohu nahlížet buď jako na maticový součin dvou vektorů:

$$(30, 45) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = 30x_1 + 45x_2$$

nebo jako na skalární součin dvou vektorů:

$$(30, 45)(x_1, x_2) = 30x_1 + 45x_2.$$

7.4 Simplexová metoda

Simplexová metoda neboli simplexový algoritmus je základní univerzální iterativní způsob řešení problémů lineárního programování, který byl v roce 1947 objeven americkým matematikem Georgem Dantzigem. Vychází z Gaussovy metody úplné eliminace, která je upravena tak, že v každé iteraci vzrostla (maximalizační úloha), popřípadě klesla (minimalizační úloha) hodnota účelové funkce, a přitom pravé strany u omezujících podmínek zůstaly nezáporné (Hliněný, 2007).

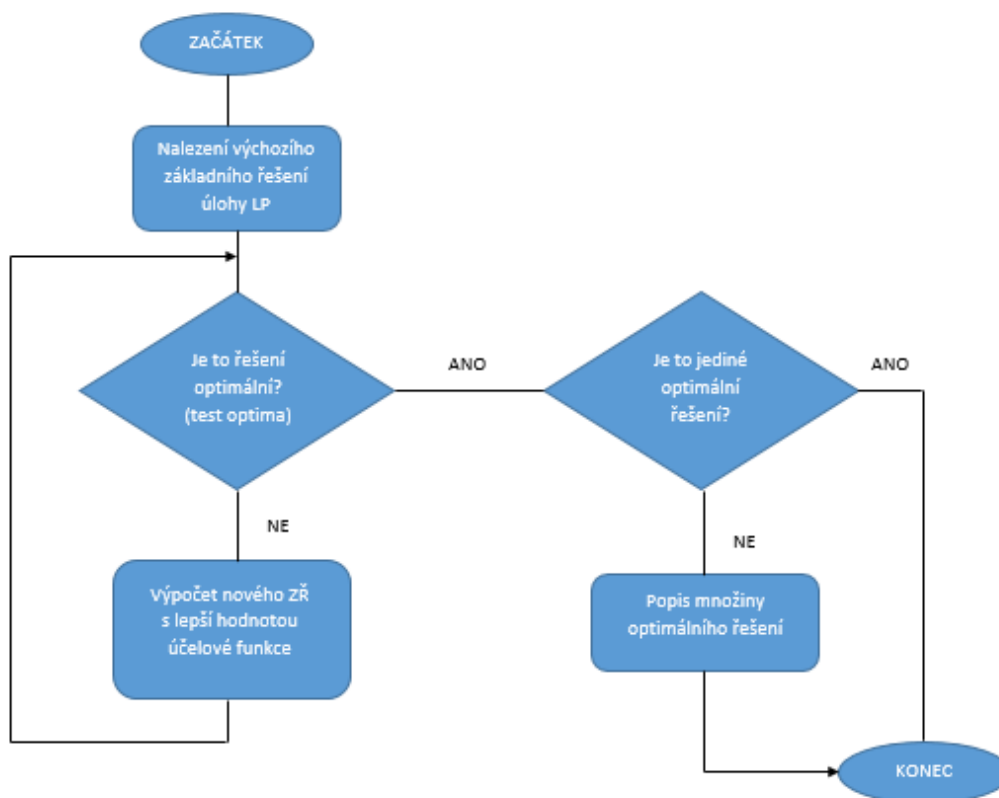
Principem této metody je podle základní věty LP hledání optimálního řešení mezi konečným počtem základních řešení. Nejprve nalezneme základní řešení, od kterého postupujeme dále a v každém dalším kroku se řešení pozmění takovým způsobem, aby hodnota účelové funkce byla vyšší než v kroku předchozím. Simplexová metoda musí poskytnout návod, který zaručí, že se postup nebude opakovat a v konečném počtu kroků dojde k závěru. Průběh metody je ukončen ve chvíli, kdy již výsledné

řešení nelze zlepšit, je tedy považováno za optimální (Lagová a Jablonský, 1999) (Simplexová metoda, 2016).

Postup výpočtu pomocí simplexové metody můžeme rozdělit na dvě základní fáze.

1. Fáze: Nalezení výchozího základního řešení
2. Fáze: Iterační postup vedoucí k optimalizaci účelové funkce z.

V některých případech nalezneme základní řešení tak snadno, že 1. Fáze výpočtu odpadá, takový případ označujeme jako jednofázovou simplexovou metodu. V jiných případech však nemusí být nalezení výchozího základního řešení vůbec jednoduché nebo nemusí vůbec existovat, v tomto případě se jedná o dvofázovou simplexovou metodu.



Obrázek 14: Hrubé schéma simplexové metody

Vlastní zpracování, zdroj: Jablonský, J., 2007

Základní postup řešení problémů lineárního programování pomocí simplexové metody vyžaduje omezení ve tvaru nerovností a všechny proměnné, vyskytující se v modelu, musí být kladné. Tento požadavek je splněn za pomoci vlastních omezujících podmínek a omezujících podmínek nezápornosti, které jsou definovány v rámci

matematického modelu. Hrubé schéma simplexové metody je popsáno na nákresu (Obrázek 14).

7.4.1 Jednofázová simplexová metoda

Pro názornou ukázkou fungování této metody použijeme vzorový příklad prodeje dvou druhů bonboniér s maximalizací zisku, který jsem popisovala v předchozích podkapitolách. Jednofázovou simplexovou metodu můžeme použít pouze v případě, že všechna vlastní omezení úlohy LP jsou definována jako nerovnice typu „ $\leq$ “, stejně jako v mém ilustračním případu (Ferguson, 2015).

Máme následující soustavu nerovnic:

$$2x_1 + 10x_2 \leq 60$$

$$6x_1 + 6x_2 \leq 60$$

$$10x_1 + 5x_2 \leq 85$$

Tuto soustavu nerovnic je třeba převést na ekvivalentní soustavu rovnic pomocí tzv. přídatných proměnných. Dostaneme tak soustavu rovnic ve speciálním tvaru, který nám usnadní získání výchozího základního řešení. Jedná se o soustavu rovnic, ve kterých obsahuje matice strukturních koeficientů m jednotkových sloupcových vektorů, které můžeme uspořádat do jednotkové matice. Tento tvar soustavy rovnic označujeme jako **kanonický tvar**. Ze soustavy m lineárních rovnic s $(m + n)$ počtem proměnných v kanonickém tvaru můžeme snadno odvodit základní řešení této soustavy (Kalčevová, 2017).

Prvním krokem v postupu jednofázové simplexové metody je odstranění nerovností ze vzorců. Vznikne následující soustava rovnic, kde proměnné x_3 , x_4 a x_5 jsou základní proměnné a x_1 , x_2 jsou proměnné přídatné:

$$2x_1 + 10x_2 + x_3 = 60$$

$$6x_1 + 6x_2 + x_4 = 60$$

$$10x_1 + 5x_2 + x_5 = 85$$

Odpovídající matice strukturních koeficientů

$$\begin{bmatrix} 2 & 10 & 1 & 0 & 0 \\ 6 & 6 & 0 & 1 & 0 \\ 10 & 5 & 0 & 0 & 1 \end{bmatrix}$$

Tato matice obsahuje tři jednotkové vektory, které lze uspořádat do jednotkové matice. Pokud položíme nezákladní proměnné rovny 0, potom z dané soustavy rovnic získáme hodnoty základních proměnných, tedy:

$$x_3 = 60, x_4 = 60, x_5 = 85$$

Vzhledem k nezápornosti všech složek tohoto řešení, považujeme toto řešení za základní.

Poté, co získáme základní řešení můžeme přistoupit k provádění iterací a k testu optimality, čímž budeme zlepšovat stávající řešení. Tyto výpočty jsou organizovány v tzv. simplexové tabulce (Tabulka 25).

Tabulka 25: Výchozí simplexová tabulka

Zákl.prom.	X₁	X₂	X₃	X₄	X₅	b	t
X ₃	2	10	1	0	0	60	6
X ₄	6	6	0	1	0	60	10
X ₅	10	5	0	0	1	85	17
Z_j	-35	-45	0	0	0	0	

Sloupce v této tabulce odpovídají jednotlivým proměnným v modelu a řádky tabulky vyznačují jednotlivé omezující podmínky. V posledním řádku tabulky je zápis účelové funkce. Anulovaný tvar účelové funkce poté vypadá následovně:

$$Z - 35x_1 - 45x_2 = 0$$

Přepočet simplexové tabulky

Vstupující proměnná zde v simplexové tabulce určuje tzv. **klíčový sloupec** (vyznačen žlutě) a vystupující proměnná tzv. **klíčový řádek**. Průsečík tohoto klíčového sloupce a řádku nazýváme **klíčový prvek** (Tabulka 26). Pro přepočet simplexové tabulky použijeme Gaussovu eliminační metodu. Ve sloupci vstupující proměnné poté vznikne jednotkový vektor, kde jednotka bude právě na místě klíčového prvku. Zároveň zde musí být zachovány jednotkové vektory i u dalších základních proměnných. Vlastní přepočet tabulky dále probíhá následovně:

1. Transformace klíčového řádku – celý řádek vždy vydělíme klíčovým prvkem, čímž získáme hodnotu 1 na místě klíčového prvku.
2. Transformace *i*-tého řádku simplexové tabulky – klíčový řádek po transformaci (obsahuje hodnotu 1 na místě klíčového prvku) vynásobíme hodnotou $(-\alpha_{ik})$, což je záporná hodnota vstupující proměnné, a poté přičteme k *i*-tému řádku. Stejný postup použijeme i u řádku účelové funkce (Matoušek, 2006).

Tabulka 26: Výchozí simplexová tabulka s vyznačeným klíčovým sloupcem a řádkem

Zákl.prom.	$X_1 = \alpha_{ik}$	X_2	X_3	X_4	X_5	β_i	$T = \beta_i / \alpha_{ik}$
X_3	2	10	1	0	0	60	30
X_4	6	6	0	1	0	60	10
X_5	10	5	0	0	1	85	8,5
Z_j	-35	-45	0	0	0	0	

Při výpočtu postupujeme následovně:

Klíčový řádek

1. Řádek (tab. 26) = 1. řádek (tab. 25) / 10

$$X_1 = 5/10 = 1/5; X_2 = 10/10 = 1; X_3 = 1/10; X_4 = 0/10; X_5 = 0/10; \beta_i = 60/10$$

2. Řádek (tab. 26) = 2. řádek (tab. 25) + 1. řádek (tab. 26) * $(-\alpha_{ik})$

$$X_1 = 6 + (1/5 * (-6)) = 24/5; X_2 = 6 + (1 * (-6)) = 0; X_3 = 0 + (1/10 * (-6)) = -3/5; X_4 = 1 + (0 * (-6)) = 1; X_5 = 0 + (0 * (-6)) = 0; \beta_i = 60 + (6 * (-6)) = 24$$

Tímto způsobem přepočítáme všechny řádky v tabulce, včetně řádku účelové funkce, jak je možné vidět níže v tabulce (Tabulka 27).

Tabulka 27: První krok výpočtu simplexovou metodou

Zákl.prom.	X_1	X_2	X_3	X_4	X_5	β_i	$T = \beta_i / \alpha_{ik}$
X_2	1/5	1	1/10	0	0	6	30
X_4	24/5	0	-3/5	1	0	24	5
X_5	9	0	-1/2	0	1	55	55/9
Z_j	-26	0	9/2	0	0	270	

Řešení je optimální, jestliže jsou při:

- Maximalizaci účelové funkce všechny redukované ceny nezáporné

$$Z_k \geq 0, k \in N,$$

- Minimalizaci účelové funkce všechny redukované ceny nekladné

$$Z_k \leq 0, k \in N,$$

(Kde N je množina indexů nezákladních proměnných)

Pokud nelze nalézt v daném kroku výpočtu vstupující proměnnou, která by vedla ke zvýšení (u maximalizace) nebo ke snížení (u minimalizace) hodnoty účelové funkce, potom základní řešení obsažené v tomto kroku výpočtu je řešením optimálním (Vanderbei, 2006).

Pokud je však v nějakém kroku výpočtu porušen test optimality, znamená to, že lze nalézt jiné základní řešení, jehož účelová funkce bude dosahovat lepších hodnot.

Jelikož zde platí pravidlo, že v účelové funkci musí být ve výsledném řešení pouze nezáporná reálná čísla, tak musím provést úpravu simplexové tabulky ještě jednou. Postupujeme stejně jako v přechozím případě, s tím rozdílem, že se mění umístění klíčového řádku a klíčového sloupce (Lineární programování a řešení některých úloh simplexovou metodou, 2012).

Tabulka 28: Optimální řešení úlohy lineárního programování

Zákl.prom.	X₁	X₂	X₃	X₄	X₅	β_i
X ₁	1	1	1/8	-1/24	0	5
X ₂	0	0	-1/8	5/24	0	5
X ₅	0	0	5/8	-15/8	1	10
Z_j	0	0	5/4	65/12	0	400

Ve výše uvedené tabulce (Tabulka 28) můžeme vidět výsledné řešení pro zadanou úlohu.

Optimální řešení:

$$\mathbf{X} = (5, 5, 0, 0, 10)$$

$$\mathbf{Z} = 400$$

Dospěla jsem k závěru, že optimálním řešením vzorové jednoduché úlohy na výrobu dvou druhů čokoládových bonboniér je výroba 5 ks bonboniéry 1. typu a 5 ks bonboniéry 2. typu. Při tomto řešení dojde k maximalizaci zisku z prodeje těchto bonboniér (400 Kč), a to při maximálním využití všech zdrojů (v tomto případě čokoládových bonbónů v příchutích: čokoláda 60 ks, oříšek 60 ks a karamel 85 ks) (Jablonský, 2007).

8 Popis společnosti XY

Společnost XY se pohybuje na trhu kovo zpracujících firem od roku 1992. Zaměřuje se na výrobu široké škály výlisků, výtažků a svařenců z nerez, pozinkovaného plechu, oceli, mosazi a hliníku. Mimo jiné se zabývá také vývojem nových technologií. Pro své zákazníky je schopna na zakázku navrhnout a vyrobit téměř jakýkoliv kovový dílec. Upravuje tedy lisovací nástroje a podobu a funkčnost výrobků zákazníkovi na míru.

Tato výrobní společnost se již několik let zabývá optimalizací a zvyšováním efektivnosti ve výrobě. V roce 2010 zde byl poprvé upraven layout výrobních prostor, který se od té doby několikrát změnil, mimo jiné je zde také snaha o optimalizaci skladových prostor a velikosti a počtu objednávek materiálu. Mým řešením problému sledu výrobních zakázek bych chtěla přispět ke zlepšení celkového průchodu zakázky výrobním procesem, a tím ke snížení, jak časových, tak finančních nákladů společnosti.

8.1 Současná situace ve společnosti XY

V současné době společnost XY disponuje 17 funkčními lisovacími stroji. Tyto lisy mají výměnné nástavné hlavice a lze je uzpůsobovat pro výrobu různých výrobků. Přenastavení lisu trvá v průměru 30 minut. Ve společnosti pracuje cca 53 zaměstnanců, z toho 5 administrativních pracovníků, 2 vedoucí směn a 4 mistrové dílen. Celkem je zde 40 pracovníků ve výrobě. Je zde zavedena dvousměnná výroba, jedna směna trvá 8 pracovních hodin, to znamená, že celkem výroba funguje 16 pracovních hodin denně. Výpočet pracovních hodin za týden je uveden v následující tabulce (Tabulka 29).

Tabulka 29: Výpočet pracovních hodin

Výpočet pracovních hodin	
Směna 1	8 hodin
Směna 2	8 hodin
Pracovní dny	5
Stroje	17
Celkem hodin	$[(8+8) * 5] = 80$

Celkem hodin na jednotlivých stro- jích	$[(8+8) * 5] * 17 = 1360$
--	---------------------------

Je patrné, že pouze část výrobních pracovníků z celkového počtu 40 pracuje s lisovacími stroji. Celkem pracuje 34 zaměstnanců u lisů a zbytek je ve skladovacích prostorech, u balení zboží, či v kontrole jakosti.

Stroje a lidé při plném vytížení pracují 1360 hodin za týden. Ve společnosti je doposud pro řazení zakázek ve výrobě využíván systém FCFN. Domnívám se, že tento zvolený způsob rozvrhování výroby je neefektivní a budu se snažit navrhnout lepší metodu pro řešení tohoto problému. Stávající výrobní plán je vyobrazen v následující tabulce (Tabulka 30), zakázky jsou zde řazeny podle principu First come, first next, to znamená, že jsou umísťovány na stroje v takovém pořadí, v jakém jsou přijímány od zákazníka. Zakázky jsou zpracovávány vždy celé, nejsou děleny na jednotlivé operace a stroje jsou tedy seřizeny vždy po skončení každé operace. Jak je zde možné vidět, tak týdenní lhůta na zpracování zakázky je zde překročena hned ve třech případech.

Tabulka 30: Stávající výrobní plán

	1.zakázka	2.zakázka	Celkem hodin
M1	J1/13	J18/26	39
M2	J2/39		39
M3	J3/35	J24/27	62
M4	J4/45		45
M5	J5/83		83
M6	J6/26	J23/18	44
M7	J7/20	J20/46	66
M8	J8/79		79
M9	J9/35	J25/45	80
M10	J10/15	J19/19	34
M11	J11/35	J26/73	108
M12	J12/43		43
M13	J13/20	J21/81	101
M14	J14/35		35
M15	J15/21	J22/18	39
M16	J16/40		40
M17	J17/38		38

8.2 Zdrojová data

Společnost XY mi pro zpracování této diplomové práce poskytla výrobní data, parametry produktů a informace o jednotlivých zakázkách. Konkrétně zde budu analyzovat data o zakázkách, které mají být zpracovány v průběhu 47.týdne v roce 2016. Obdržená data byla nepatrně pozměněna a upravena tak, aby zde nebyly vyraženy podstatné výrobní informace společnosti.

Zdrojový datový soubor obsahuje termín, do kterého musí být daná objednávka odeslána zákazníkovi. Dále je zde název, neboli přiřazené ID produktu. V dokumentu se některé zakázky pro výrobu určitého produktu opakují. Klient zřejmě objednává od společnosti XY pravidelně řadu stejných výrobků, termíny zpracování jednotlivých zakázek se však liší. Ve zdrojovém dokumentu samozřejmě nesmí chybět počet objednaných kusů produktu v zakázce a velmi důležitou částí tabulky jsou sloupce s informacemi o počtu operací, kterými daná zakázka prochází a dále stroje, na kterých může být daná operace uskutečněna. Podstatné jsou zde časy, za které je daná operace dané zakázky zpracována. Tyto časy jsou vypočítány z normy, která je u každého výrobku předem stanovená. Tato norma představuje optimální počet výrobků určitého typu výrobku zpracovaného za směnu. Tento počet výrobků jsem následně přepočítala a zjistila průměrný čas na jeden výrobek. Toto číslo je poté vynásobeno počtem kusů výrobku v zakázce. Výsledkem je doba strávená například první operací (ražení z kovových pásů) u dané zakázky, která obsahuje 50000 kusů kovového výlisek, tento výlisek se následně ohýbá a brousí. Časy všech těchto operací jsou dále nasčítány ve sloupci „Celkem hodin“.

Celkový počet hodin pro zpracování zadaných zakázek ve 47. týdnu je 957 hodin. Tento čas je čistý čas práce na stroji, to znamená, že v tomto počtu hodin nejsou zaneseny prostoje, seřizování strojů, kontrola, přesuny materiálu či jiné manipulace s výrobky. Cílem mé práce je minimalizace celkové doby zpracování zakázek, to znamená jednak optimalizaci sledu výroby zakázek tak, aby došlo k potřebné vytíženosti strojů a co nejmenšímu počtu seřizování lisů, a jednak minimalizaci výše zmíněných vedlejších vlivů, které značně prodlužují dobu práce na zakázkách. Původní datový soubor je uveden v přílohách (Příloha 1).

8.2.1 Úprava dat

Původní zdrojová data jsou velmi komplexní a poměrně dost obsáhlá, práce s nimi by byla velmi složitá a časově náročná. Všechny vstupující proměnné tedy není při práci se sofistikovaným softwarem možno zohlednit. Jednou z možností je optimalizace pracovního rozvrhu podle pevného data pro odevzdání jednotlivých zakázek zákazníkovi, další z možností je obecnější optimalizace plánu pro zkrácení celkové

doby zhotovení zakázek. Jako hlavní faktor pro optimalizaci výrobního plánu jsem si zvolila nejkratší čas zhotovení všech zakázek, proto mohu tzv. deadliney pro jednotlivé zakázky z datového souboru odstranit. Nejdůležitějšími proměnnými v mém datovém souboru jsou tedy pracovní časy pro jednotlivé operace v každé zakázce a zároveň návaznosti jednotlivých zakázek mezi sebou. Upravený datový soubor můžete vidět v tabulce níže (Tabulka 31).

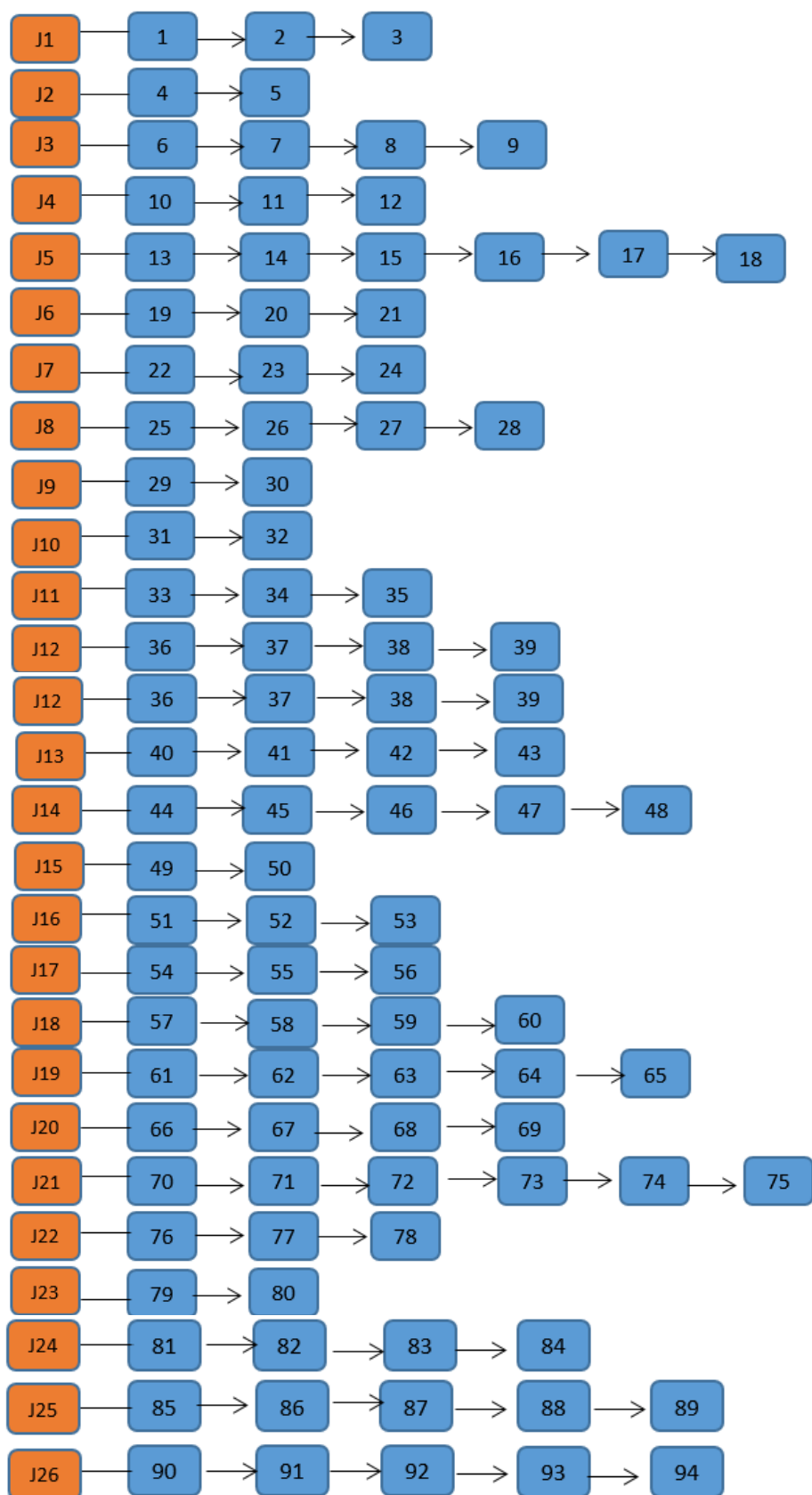
Tabulka 31: Výchozí datový soubor

#	Zakázka	Operace	Doba	Předchozí operace	Návaznosti2	Návaznosti1
1	J1	J(1,1)	5	x	-	2
2		J(1,2)	2	J(1,1)	1	3
3		J(1,3)	6	J(1,2)	2	-
4	J2	J(2,1)	21	x	-	5
5		J(2,2)	18	J(2,1)	4	-
6	J3	J(3,1)	7	x	-	7
7		J(3,2)	8	J(3,1)	6	8
8		J(3,3)	12	J(3,2)	7	9
9		J(3,4)	8	J(3,3)	8	-
10	J4	J(4,1)	17	x	-	11
11		J(4,2)	5	J(4,1)	10	12
12		J(4,3)	5	J(4,2)	11	-
13	J5	J(5,1)	18	x	-	14
14		J(5,2)	12	J(5,1)	13	15
15		J(5,3)	15	J(5,2)	14	16
16		J(5,4)	10	J(5,3)	15	17
17		J(5,5)	8	J(5,4)	16	18
18		J(5,6)	20	J(5,5)	17	-
19	J6	J(6,1)	13	x	-	20
20		J(6,2)	8	J(6,1)	19	21
21		J(6,3)	5	J(6,2)	20	-
22	J7	J(7,1)	7	x	-	23
23		J(7,2)	8	J(7,1)	22	24
24		J(7,3)	5	J(7,2)	23	-
25	J8	J(8,1)	21	x	-	26
26		J(8,2)	24	J(8,1)	25	27
27		J(8,3)	15	J(8,2)	26	28
28		J(8,4)	19	J(8,3)	27	-
29	J9	J(9,1)	25	x	-	30
30		J(9,2)	10	J(9,1)	29	-
31	J10	J(10,1)	3	x	-	32

32		J(10,2)	12	J(10,1)	31	-	
33	J11	J(11,1)	10	x	-		34
34		J(11,2)	15	J(11,1)	33		35
35		J(11,3)	10	J(11,2)	34	-	
36	J12	J(12,1)	10	x	-		37
37		J(12,2)	14	J(12,1)	36		38
38		J(12,3)	12	J(12,2)	37		39
39		J(12,4)	7	J(12,3)	38	-	
40	J13	J(13,1)	3	x	-		41
41		J(13,2)	3	J(13,1)	40		42
42		J(13,3)	7	J(13,2)	41		43
43		J(13,4)	7	J(13,3)	42	-	
44	J14	J(14,1)	2	x	-		45
45		J(14,2)	3	J(14,1)	44		46
46		J(14,3)	8	J(14,2)	45		47
47		J(14,4)	4	J(14,3)	46		48
48		J(14,5)	18	J(14,4)	47	-	
49	J15	J(15,1)	14	x	-		50
50		J(15,2)	7	J(15,1)	49	-	
51	J16	J(16,1)	13	x	-		52
52		J(16,2)	18	J(16,1)	51		53
53		J(16,3)	9	J(16,2)	52	-	
54	J17	J(17,1)	12	x	-		55
55		J(17,2)	18	J(17,1)	54		56
56		J(17,3)	8	J(17,2)	55	-	
57	J18	J(18,1)	9	x	-		58
58		J(18,2)	4	J(18,1)	57		59
59		J(18,3)	5	J(18,2)	58		60
60		J(18,4)	8	J(18,3)	59	-	
61	J19	J(19,1)	1	x	-		62
62		J(19,2)	4	J(19,1)	61		63
63		J(19,3)	9	J(19,2)	62		64
64		J(19,4)	3	J(19,3)	63		65
65		J(19,5)	2	J(19,4)	64	-	
66	J20	J(20,1)	25	x	-		67
67		J(20,2)	3	J(20,1)	66		68
68		J(20,3)	5	J(20,2)	67		69
69		J(20,4)	13	J(20,3)	68	-	
70	J21	J(21,1)	15	x	-		71
71		J(21,2)	8	J(21,1)	70		72
72		J(21,3)	24	J(21,2)	71		73

73		J(21,4)	4	J(21,3)	72	74
74		J(21,5)	8	J(21,4)	73	75
75		J(21,6)	22	J(21,5)	74	-
76	J22	J(22,1)	4	x	-	77
77		J(22,2)	5	J(22,1)	76	78
78		J(22,3)	9	J(22,2)	77	-
79	J23	J(23,1)	13	x	-	80
80		J(23,2)	5	J(23,1)	79	-
81	J24	J(24,1)	2	x	-	82
82		J(24,2)	10	J(24,1)	81	83
83		J(24,3)	10	J(24,2)	82	84
84		J(24,4)	5	J(24,3)	83	-
85	J25	J(25,1)	3	x	-	86
86		J(25,2)	4	J(25,1)	85	87
87		J(25,3)	14	J(25,2)	86	88
88		J(25,4)	17	J(25,3)	87	89
89		J(25,5)	7	J(25,4)	88	-
90	J26	J(26,1)	6	x	-	91
91		J(26,2)	19	J(26,1)	90	92
92		J(26,3)	13	J(26,2)	91	93
93		J(26,4)	14	J(26,3)	92	94
94		J(26,5)	21	J(26,4)	93	-
		SUM	957		68	68

Pro lepší práci s daty jsem každou zakázku rozdělila na jednotlivé operace a definovala návaznosti mezi nimi, které jsou popsány ve sloupcích *předchozí operace*, *návaznosti1* a *návaznosti2*. Pro lepší popis zakázek ke zpracování uvádím následující schéma, které názorně ukazuje počet operací v jednotlivých zakázkách i jejich návaznosti. Počet operací v jednotlivých zakázkách a návaznosti mezi nimi jsou dobře vidět na následujícím schématu výrobního plánu (Obrázek 15)



Obrázek 15: Schéma zakázek ve výrobě

9 Řešení pomocí genetických algoritmů

Řešení zadané úlohy pomocí genetických algoritmů jsem se rozhodla aplikovat v programu R. R je jazyk a prostředí pro statistické výpočty a grafiku. Program R poskytuje širokou škálu statistických (lineární a nelineární modelování, klasické statistické testy, analýza časových řad, klasifikace, shlukování, atd.) a grafické techniky a poskytuje také široké možnosti rozšíření. Jazyk R se velmi podobá S jazyku, avšak jazyk S často slouží jako pouhý vyhledávací prostředek pro výzkum ve statistické metodologii, zatímco jazyk R poskytuje přímo cestu otevřeného zdroje k účasti na této činnosti (What is R?, 2015). Velkou výhodou pro mě u programu R je, že tento software je ke stažení zdarma, zatímco program MATLAB, který jsem použila pro tvorbu modelu lineárního programování se řadí mezi komerční software (Chipperfield, Fleming, Pohlheim, a Fonseca, 2014).

9.1 Ukázkový příklad

Aplikace genetických algoritmů je poměrně hodně složitá. Pokusím se tedy tuto metodu aplikovat nejdříve na jednodušším příkladu, kde máme celkem 3 zakázky, které se skládají z 8 operací. Tyto zakázky je třeba zpracovat na 3 strojích, a to v co nejkratším celkovém čase. Pomocí následujícího kódu systém automaticky vytvoří populaci, která obsahuje 200 různých řešení. Tyto řešení jsou posléze nakombinována na základě ohodnocení jejich kvality (fitness). Ze vzniklých řešení je vybráno to nejlepší, v tomto případě s nejkratší dobou rozvrhu. Jelikož je vygenerovaný jen určitý počet řešení, není zde zaručeno nalezení optimálního řešení, ale pouze řešení suboptimálního (Cheung a Zhou, 2001).

V první části kódu nejdříve zadáváme pracovní oblast, kde se budeme pohybovat, a sice knihovnu pro práci s genetickými algoritmy. Dále zde zadáváme počet operací (8) a počet strojů, které máme k dispozici (3). V další části popisujeme datový soubor jako takový, kde jsou jednotlivé operace rozděleny do třech zakázek. Řádek *job* udává znovu rozdělení operací do jednotlivých zakázek, podobně jako řádek *subjob*. *Duration* vyznačuje doby trvání jednotlivých operací a *devec2* definuje návaznosti operací v jakém pořadí musí jít po sobě.

```
library(genalg)
subjobs <- 8
m <- 3
dataset<-data.frame(operation=c("J(1,1)", "J(1,2)", "J(1,3)", "J(2,1)", "J(2,2)", "J(3,1)",
                                "J(3,2)", "J(3,3)"),
```

```
job = c(1,1,1,2,2,3,3,3),  
subjob = c(1,2,3,1,2,1,2,3),  
duration = c(5,2,6,21,18,7,8,12),  
depvec2 = c(0,1,2,0,4,0,6,7))
```

Maximální doba trvání prací na všech zakázkách má být menší nebo rovna součtu časů jednotlivých operací.

```
max_duration <- sum(dataset$duration)
```

Nyní definujeme účelovou funkci jako základ pro ohodnocení fitness, a sice že minimální trvání zhotovení celého rozvrhu je maximum ze součtu časů na jednotlivých strojích.

```
evalFunc <- function(x) {  
  mat <- matrix(x, ncol=3, byrow=T)  
  current_duration <- max(apply(dataset$duration * mat, 2, sum))
```

Zde zavádíme první omezující podmínku, která znamená, že každý úkol je přiřazen právě na jeden stroj.

```
#podmínka: all(apply(mat, 1, sum) == 1)
```

Druhá podmínka znamená, že jednotlivé subjoby jsou každý přiřazen jinému stroji (tato podmínka by zde nemusela být, pokud má každá zakázka zcela jiný průběh).

```
#podmínka: all(apply(mat, 2, tapply, dataset$job, sum) <= 1)  
  if (current_duration > max_duration) return(max_duration)  
  else  
    if (any(apply(mat, 1, sum) != 1)) return(max_duration)  
  else  
    if (any(apply(mat, 2, tapply, dataset$job, sum) >  
1)) return(max_duration)  
    else return(current_duration)}
```

Následuje zápis pro samotný výpočet řešení, a tedy, že počáteční populace obsahuje 2000 různých scénářů, počet iterací je 200 a šance na mutaci je 0,05.

```
iter = 200  
GAmode1 <- rbga.bin(size = m*subjobs, popSize = 2000, iters  
= iter, mutationChance = 0.05, elitism = T, evalFunc = eval-  
Func, verbose=F)  
GAmode1$best  
cat(summary(GAmode1))
```

9.2 Výsledné řešení

V první části výsledného řešení je ukázáno, jak se s vyšším počtem vygenerovaných řešení nachází čím dál tím lepší řešení. Můžeme tedy usuzovat, že se zvyšujícím se počtem iterací se zvyšuje přesnost řešení a je větší šance pro nalezení lepšího suboptimálního řešení. Výsledné řešení bylo nalezeno při 189. iteraci a nejkratší možný čas zhotovení všech zakázek je 39 hodin.

```
evalFunc, verbose=F)
> GAmodel$best
[1] 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79
79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79 79
79 79 79 79 79 79 79 79 79
[48] 79 79 79 79 79 79 79 79 79 79 79 79 79 79 39 39 39 39 39
39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
39 39 39 39 39 39 39 39 39
[95] 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
39 39 39 39 39 39 39 39 39
[142] 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39 39
39 39 39 39 39 39 39 39 39
[189] 39 39 39 39 39 39 39 39 39 39 39 39 39
```

Zde jsou znovu zopakovány nastavené parametry pro hledání suboptimálního řešení.

```
> cat(summary(GAmodel))
GA Settings
Type = binary chromosome
Population size = 2000
Number of Generations = 200
Elitism = TRUE
Mutation Chance = 0.05

Search Domain
Var 1 = [,]
Var 0 = [,]
```

A konečně je zde výsledné řešení, které je uvedeno pomocí binárního kódu. Tento kód přeloží tak, že každé tři znaky říkají, na kterém stroji má být daná operace zpracována. Tedy pokud je zde trojčíslí 001, znamená to, že operace bude probíhat na stroji číslo 3, kód 010 znamená využití stroje číslo 2 a číslo 100 určuje operaci na stroj číslo 1.

GA Results

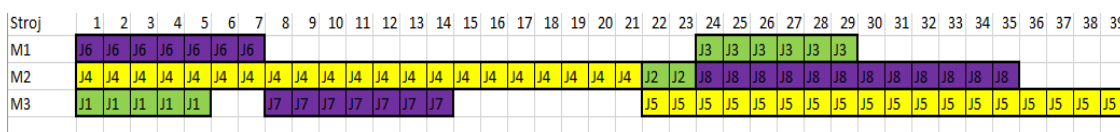
Best Solution : 0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0

Binární kód rozdělíme následovně, jak je vidět, tak nám vzniklo celkem 8 oddělených částí, které reprezentují 8 zadaných operací (subjobů).

0 0 1 | 0 1 0 | 1 0 0 | 0 1 0 | 0 0 1 | 1 0 0 | 0 0 1 | 0 1 0

9.3 Interpretace výsledků

Pro lepší přehlednost jsem ze získaného řešení vytvořila Ganttův diagram (Obrázek 16), který potvrzuje, že nalezené řešení je správné a celý pracovní rozvrh je opravdu možné stihnout za 39 hodin, a to včetně čekacích dob.



Obrázek 16: Ganttův diagram LP

I toto řešení má však svoji nevýhodu, a sice, že takto vygenerovaných řešení jsem měla několik a jejich výsledky byly různé. Systém vygeneruje první suboptimální řešení, které splňuje zadané podmínky, ovšem ne v každém řešení byla dodržena návaznost mezi operacemi. Pro splnění podmínky zachování vztahů návaznosti by bylo nutné kód doprogramovat, což by bylo ovšem dost obtížné.

Tabulka 32: Pokusy řešení metodou LP

Pokus	Řešení	Počet hodin
1.pokus	0 0 1 0 1 0 0 1 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0	35
2.pokus	0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0	34
3.pokus	1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 0 1 0 1 0	37
4.pokus	0 0 1 0 1 0 1 0 0 0 1 0 0 0 1 1 0 0 0 0 1 0 1 0	39
5.pokus	0 0 1 0 1 0 1 0 0 1 0 0 0 1 0 1 0 0 0 0 1 0 1 0	34

Jak můžeme vidět ve výše uvedené tabulce (Tabulka 32), tak pouze jedno z 5 výsledných řešení je proveditelné v daném čase. Pomocí metody genetických algoritmů tedy lze dospět k uspokojivému suboptimálnímu řešení, avšak i u takto jednoduché implementace je třeba vygenerovat velké množství řešení a z nich následně ručně vybírat. Nevýhodou tohoto způsobu řešení rozvrhovacího problému je také dlouhý běh systému před nalezením uspokojivého řešení. Program R generuje a propočítává různé varianty a vznik každého řešení trvá i několik minut i u takto jednoduchého zadání jako byl můj vzorový příklad. Tento problém bych mohla vyřešit sofistikovanějším zakódováním, které by zahrnovalo i vazby v problému, to by bylo však již nad rámec diplomové práce.

9.4 Kód pro zadaný případ

Po zpracování jednoduchého příkladu v jazyku R přistoupím k aplikaci tohoto postupu na zadaný příklad reálného stavu výroby ve společnosti XY. Následující kód je velmi podobný jako kód u vzorového příkladu, pouze jsou zde dosazena data reálných zakázek a strojů. Tento kód tedy nebudu podrobněji komentovat.

```
library(genalg)
subjobs <- 94
m <- 17
dataset<-data.frame(operation=c("J(1,1)", "J(1,2)", "J(1,3)", "J(2,1)", "J(2,2)", "J(3,1)",
"J(3,2)", "J(3,3)", "J(3,4)", "J(4,1)", "J(4,2)", "J(4,3)", "J(5,1)", "J(5,2)", "J(5,3)", "J(5,4)", "J(5,5)", "J(5,6)", "J(6,1)",
"J(6,2)", "J(6,3)", "J(7,1)", "J(7,2)", "J(7,3)", "J(8,1)", "J(8,2)", "J(8,3)", "J(8,4)", "J(9,1)", "J(9,2)", "J(10,1)", "J(10,2)",
"J(11,1)", "J(11,2)", "J(11,3)", "J(12,1)", "J(12,2)", "J(12,3)", "J(12,4)", "J(13,1)", "J(13,2)", "J(13,3)", "J(13,4)", "J(14,1)", "J(14,2)", "J(14,3)", "J(14,4)", "J(14,5)", "J(15,1)", "J(15,2)", "J(16,1)", "J(16,2)", "J(16,3)", "J(17,1)", "J(17,2)", "J(17,3)", "J(18,1)", "J(18,2)", "J(18,3)", "J(18,4)", "J(19,1)", "J(19,2)", "J(19,3)", "J(19,4)", "J(19,5)", "J(20,1)", "J(20,2)", "J(20,3)", "J(20,4)", "J(21,1)", "J(21,2)", "J(21,3)", "J(21,4)", "J(21,5)", "J(21,6)", "J(22,1)", "J(22,2)", "J(22,3)", "J(23,1)", "J(23,2)", "J(24,1)", "J(24,2)", "J(24,3)", "J(24,4)", "J(25,1)", "J(25,2)", "J(25,3)", "J(25,4)", "J(25,5)", "J(26,1)", "J(26,2)", "J(26,3)", "J(26,4)", "J(26,5)"),
```

```

job =
c(1,1,1,2,2,3,3,3,3,4,4,4,5,5,5,5,5,5,6,6,6,7,7,7,8,8,8,8,9
,9,10,10,11,11,11,12,12,12,12,13,13,13,13,14,14,14,14,14,15
,15,16,16,16,17,17,17,18,18,18,18,19,19,19,19,19,20,20,20,2
0,21,21,21,21,21,21,22,22,22,23,23,24,24,24,24,25,25,25,25,
25,26,26,26,26,26),

subjob =
c(1,2,3,1,2,1,2,3,4,1,2,3,1,2,3,4,5,6,1,2,3,1,2,3,1,2,3,4,1
,2,1,2,1,2,3,1,2,3,4,1,2,3,4,1,2,3,4,5,1,2,1,2,3,1,2,3,1,2,
3,4,1,2,3,4,5,1,2,3,4,1,2,3,4,5,6,1,2,3,1,2,1,2,3,4,1,2,3,4
,5,1,2,3,4,5),

duration =
c(5,2,6,21,18,7,8,12,8,17,5,5,18,12,15,10,8,20,13,8,5,7,8,5
,21,24,15,19,25,10,3,12,10,15,10,10,14,12,7,3,3,7,7,2,3,8,4
,18,14,7,13,18,9,12,18,8,9,4,5,8,1,4,9,3,2,25,3,5,13,15,8,2
4,4,8,22,4,5,9,13,5,2,10,10,5,3,4,14,17,7,6,19,13,14,21),

depvec2 =
c(0,1,2,0,4,0,6,7,8,0,10,11,0,13,14,15,16,17,0,19,20,0,22,2
3,0,25,26,27,0,29,0,31,0,33,34,0,36,37,38,0,40,41,42,0,44,4
5,46,47,0,49,0,51,52,0,54,55,0,57,58,59,0,61,62,63,64,0,66,
67,68,0,70,71,72,73,74,0,76,77,0,79,0,81,82,83,0,85,86,87,8
8,0,90,91,92,93))

max_duration <- sum(dataset$duration)

evalFunc <- function(x) {
mat <- matrix(x, ncol=17, byrow=T)
current_duration <- max(apply(dataset$duration * mat,2,sum))
#podmínka: all(apply(mat, 1, sum) == 1)
#podmínka: all(apply(mat,2, tapply,dataset$job,sum) <= 1)
  if (current_duration > max_duration) return(max_duration)
  else
    if (any(apply(mat, 1, sum) != 1)) return(max_duration)
else
  if (any(apply(mat,2, tapply,dataset$job,sum) > 1))
return(max_duration)

```

```
        else return(current_duration) }  
iter = 200  
GAmodel <- rbga.bin(size = m*subjobs, popSize = 2000, iters  
= iter, mutationChance = 0.05, elitism = T, evalFunc = eval-  
Func, verbose=F)  
GAmodel$best  
cat(summary(GAmodel))
```

V systému R by mělo být proveditelné vygenerovat suboptimální řešení i k mému rozsáhlému případu na základě tohoto kódu. Bohužel však systém propočítává řešení příliš dlouho a k žádnému uspokojivému řešení se nedopracuje. Pravděpodobně by bylo třeba provést určité změny v kódu a doplnit podmínku pro lepší respektování vztahů a návazností mezi jednotlivými operacemi. Detailnější a složitější zakódování problému profesionálním softwaru by bylo nad rámec diplomové práce. Cíle práce jsou naplněny i bez takto sofistikované implementace.

10 Řešení pomocí lineárního programování

K řešení zadaného příkladu metodou lineárního programování jsem použila software MATLAB s využitím doplňku TOMLAB. MATLAB je jeden z nejpoužívanějších programů v oblastech strojového učení, počítačové vizualizace, finanční výpočetnictví a modelování, dále různé počítačové vizualizace a zpracování procesů. Rozhraní MATLAB je optimalizováno pro řešení pokročilých a velmi specifických problémů (Koláček a Konečná, 2014).

TOMLAB je prostředí v MATLAB, které slouží k účelovému rozvoji a modelování praktických optimalizačních úloh, dále k výzkumu, učení a řešení různých modelových situací. Optimalizační prostředí v TOMLAB je flexibilní a robustní, ale také snadno použitelné a spolehlivé pro řešení všech typů aplikovaných optimalizačních problémů. Doplňku TOMLAB dala vznik zejména potřeba po pokročilém a spolehlivém nástroji, který je použitelný pro vývoj algoritmů a softwaru pro řešení různých optimalizačních problémů v praxi (About TomLab, 2016).

Jelikož MATLAB je komerční software a licence na jeho používání je zpoplatněna, tak jsem pro účely mé diplomové práce využila pouze 30-ti denní trial verzi tohoto programu zdarma (Koláček a Konečná, 2014).

10.1 Popis kódu

V první části kódu je zadání dané úlohy, nejdříve v prvním řádku pod označením *duration* jsou vypsány doby trvání pro jednotlivé operace. Poté je zde definováno, že zadané operace mají být zhotoveny, celkem máme k dispozici 17 strojů. Další dva řádky vyjadřují návaznosti mezi jednotlivými operacemi, a sice v prvním řádku jsou ve směru od konce k začátku výrobního plánu. Je zde definováno, že určitá operace musí čekat na jinou operaci, která musí být dokončena dříve, než mohou být práce na této operaci započaty. Naopak druhý řádek popisu následnosti naopak, například, že operace 2 čeká až skončí operace 1 a zároveň operace 3 čeká na dokončení operace 2 (The MathWorks, Inc., 2017).

```
dura-  
tion=[5;2;6;21;18;7;8;12;8;17;5;5;18;12;15;10;8;20;13;8;5;7  
;8;5;21;24;15;19;25;10;3;12;10;15;10;10;14;12;7;3;3;7;7;2;3  
;8;4;18;14;7;13;18;9;12;18;8;9;4;5;8;1;4;9;3;2;25;3;5;13;15  
;8;24;4;8;22;4;5;9;13;5;2;10;10;5;3;4;14;17;7;6;19;13;14;21  
];  
stations=17;
```

```
depend-
svec1=[2;3;5;7;8;9;11;12;14;15;16;17;18;20;21;23;24;26;27;2
8;30;32;34;35;37;38;39;41;42;43;45;46;47;48;50;52;53;55;56;
58;59;60;62;63;64;65;67;68;69;71;72;73;74;75;77;78;80;82;83
;84;86;87;88;89;91;92;93;94];
depend-
svec2=[1;2;4;6;7;8;10;11;13;14;15;16;17;19;20;22;23;25;26;2
7;29;31;33;34;36;37;38;40;41;42;44;45;46;47;49;51;52;54;55;
57;58;59;61;62;63;64;66;67;68;70;71;72;73;74;76;77;79;81;82
;83;85;86;87;88;90;91;92;93];
```

V další části kódu jsou definovány další proměnné i a m , kde i je délka prací na operacích, tedy čas jejich zhotovení a m zde zastupuje počet strojů. Dále konečně definujeme účelové funkce s využitím doplňkové funkce $tom(x)$, která optimalizuje proměnnou $cycle$, což je čas zhotovení všech zakázek. Hledáme minimum této funkce.

```
i=length(duration);
m=stations;
proc=tom('proc',i,m,'int');
cycle=tom('cycle',1,1);
```

Zde jsou nastavena omezení, konkrétně podmínka nezáporného výsledku u optimalizované proměnné $cycle$.

```
%All slots are integers
bnds={0 <= proc <=1, cycle >=0};
```

V této podmínce je definováno, že každá operace může probíhat právě na jednom stroji.

```
%Every process has to go to one machine
con1={sum(proc,2) == 1};
```

V následující části kódu jsou zadány další omezující podmínky modelu, a sice podmínky návazností jednotlivých operací na sebe.

```
%Precedence constraint
iter = length(dependsvec1);
con2={};
for i=1:iter
    idxi=dependsvec1(i);
    idxj=dependsvec2(i);
    con2{i}={sum(proc(idxi,:).*(1:17)) >=...
    sum(proc(idxj,:).*(1:17))};
```

```
end  
%Cycle constraint  
con3={duration'*proc <= cycle};
```

Zde je zadán cíl účelové funkce tedy, že optimalizujeme proměnnou *cycle*.

```
%Objective  
objective=cycle;
```

Níže je shrnutí doposud zadaných proměnných do modelu a důležitý je poslední řádek, který říká, že pomocí funkce *ezsolve* minimalizujeme proměnnou *objective* s dodržением zadaných omezujících podmínek. Jak již bylo výše zmíněno, proměnná *objective* je rovna proměnné *cycle*.

```
constraints={bnds, con1, con2, con3};  
options=struct;  
options.solver='cplex';  
options.name='Assembly Line Balancing';  
sol=ezsolve(objective, constraints, [], options);
```

V poslední části kódu je zadáno, v jaké formě požadujeme výstup, tedy že přiřazujeme operace k jednotlivým strojům a generujeme různá možná řešení a z nich vybíráme to s nejmenší výslednou hodnotu proměnné *cycle*.

```
PriLev=1;  
if PriLev>0  
    t=length(duration); %number of tasks  
    s=stations; %number of workstations  
    temp=sol.proc; %reshaping distribution  
    time=sol.cycle; %total time  
  
    for i=1:s  
        disp(['tasks managed by station ' ... %filter + disp  
            num2str(i) ':' ...  
            num2str(find(temp(:,i)))'])  
    end  
    disp(['total time ' num2str(time)]) %disp the time  
end
```

(TOMLAB models manual, 2002)

10.2 Výsledné řešení

Program MATLAB v tomto případě dle mého názoru funguje trochu lépe než systém R, který jsem používala v předchozím řešení za pomoci genetických algoritmů. Ve výsledném řešení je vyznačen počet vygenerovaných řešení, v tomto případě 600, to znamená že systém vygeneroval celkem 600 možných řešení, tedy rozvrhů operací a z nich vybral to, které dosahuje nejkratšího možného času zhotovení všech zakázek. Je zde také uvedena doba výpočtu těchto řešení, což je v tomto případě 1,078 vteřiny.

Přehledně je výsledné řešení členěno podle jednotlivých strojů a ke každému z nich jsou přiřazeny operace, které se budou zpracovávat na tom daném stroji. Na základě tohoto zápisu je tedy jednoduché vytvořit Gantt chart (Ganttův diagram), který je dle mého názoru nejlepším způsobem, jak interpretovat výsledky rozvrhovacích úloh.

```
Problem: --- 1: Assembly Line Balancing
f_k      57.000000000000000000

f(x_0)    0.000000000000000000

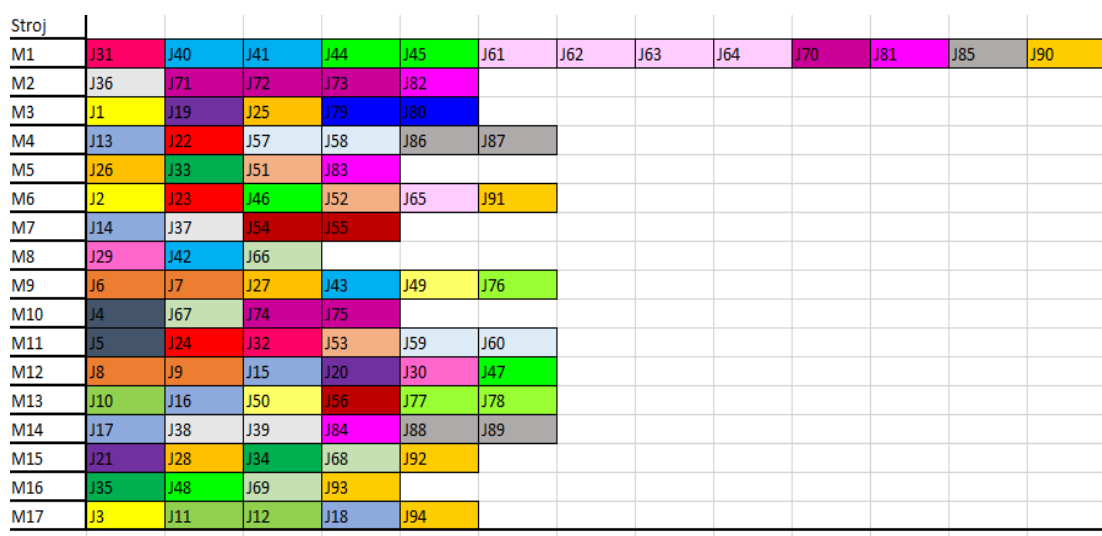
Solver: CPLEX.  EXIT=0.  INFORM=101.
CPLEX Branch-and-Cut MIP solver
Optimal integer solution found

FuncEv  600
CPU time: 1.078125 sec. Elapsed time: 0.606000 sec.
tasks managed by station 1:31  40  41  44  45  61  62  63
64  70  81  85  90
tasks managed by station 2:36  71  72  73  82
tasks managed by station 3:1  19  25  79  80
tasks managed by station 4:13  22  57  58  86  87
tasks managed by station 5:26  33  51  83
tasks managed by station 6:2  23  46  52  65  91
tasks managed by station 7:14  37  54  55
tasks managed by station 8:29  42  66
tasks managed by station 9:6   7  27  43  49  76
```

```
tasks managed by station 10:4  67  74  75
tasks managed by station 11:5  24  32  53  59  60
tasks managed by station 12:8   9  15  20  30  47
tasks managed by station 13:10  16  50  56  77  78
tasks managed by station 14:17  38  39  84  88  89
tasks managed by station 15:21  28  34  68  92
tasks managed by station 16:35  48  69  93
tasks managed by station 17:3   11  12  18  94
total time 57
```

10.3 Interpretace výsledků

Na první pohled vypadá výsledné řešení velmi dobře, systém poměrně rychle nasimuloval 600 scénářů rozvrhů výroby a z nich nakonec vybral suboptimální řešení, které dosahuje nejkratšího času zhotovení všech zakázek. Dle výsledků bych všech 94 operací mělo být zpracováno za 57 hodin. Rozvrh jednotlivých operací je zakreslen v následujícím Ganttově diagramu (Obrázek 17).



Obrázek 17: Ganttův diagram- základní řešení GA

[illegible]

Obrázek 18: Rozpracovaný Ganttův diagram

Z výše uvedeného diagramu je patrné, že výsledný rozvrh zakázek k jednotlivým strojům je reálně zpracovat za minimálně 70 hodin, nikoliv 57, jak ukázal výsledek.

V tomto řešení však nejsou ještě zahrnuty doby čekání, které vznikají kvůli zadaným návaznostem jednotlivých operací. Po zanesení těchto návaznostních vztahů do diagramu (Obrázek 19) jsem zjistila, že reálná doba zpracování zakázek je ještě delší. Tento diagram jsem tvořila ručně a je tedy možné, že existuje nějaká cesta, jak změnit pořadí zpracování těchto zakázek.

Dle tohoto diagramu je možné zpracovat výrobní plán za 145 hodin. Pomocí lineárního programování v systému MATLAB jsem tedy nedospěla k uspokojivému řešení. Jsem si však jistá, že by bylo možné tímto způsobem nalézt ještě lepší řešení, vyžadovalo by to však velké změny v programovém kódu, k čemuž bych potřebovala rozsáhlejší znalosti programovacích jazyků a prostředí MATLAB softwaru, což by již přesahovalo rámec diplomové práce.



11 Porovnání výsledků

Pro řešení složité rozvrhovací úlohy z praxe jsem použila dvě metody, jednak metodu genetických algoritmů a poté metodu lineárního programování. Pro práci s genetickými algoritmy jsem použila programovací jazyk R a řešení úlohy pomocí lineárního programování jsem vypracovala v prostředí programu MATLAB.

Metoda genetických algoritmů je poměrně složitá a vyžaduje rozsáhlé znalosti v oblasti kódování v jazyku R. Bohužel se mi nepodařilo aplikovat metodu na zadaný příklad ze výroby ve společnosti XY. Pro ukázkou řešení jsem použila jednodušší příklad s menším množstvím zakázek a strojů, i tak však software R vyžadoval poměrně hodně dlouhou dobu na výpočet řešení, i na velmi zjednodušenou variantu zadání bylo třeba několik minut na vygenerování alespoň suboptimálního řešení. Další z nevýhod řešení úlohy pomocí genetických algoritmů je složitost a nepřehlednost výsledného řešení, které je vyhotoveno v binárním kódu, toto by však šlo nastavit v kódu jinak. Domnívám se, že použití binárního kódování je vhodné pouze pro velmi jednoduché příklady, u rozsáhlejších úloh je výsledné řešení nepřehledné.

Pro řešení příkladu pomocí metody lineárního programování jsem použila kód ze zdroje TOMLAB models manual, 2002. Tento kód byl napsán pro mírně odlišný příklad, a sice řízení stavebního projektu. Kód jsem modifikovala tak, aby byl použitelný pro můj příklad, práce s kódem pro mě byla podstatně jednodušší než u předchozí varianty řešení. Software MATLAB v tomto případě vygeneroval suboptimální řešení, jehož zápis je velmi přehledný a snadno prezentovatelný. Také doba výpočtu řešení je podstatně kratší, než je tomu u metody genetických algoritmů výše. Za nevýhodu této varianty řešení bych považovala fakt, že jsem bohužel nepřišla na takový způsob, jak ideálně zapsat do kódu skutečnost, že jednotlivé operace mají mezi sebou návaznosti, které musí být při hledání řešení respektovány. Výsledné řešení tedy vychází poměrně hezky, avšak nezapočítává doby čekání na strojích mezi jednotlivými na sebe navazujícími operacemi. Tento problém je popsán názorně výše v předchozí kapitole pomocí Ganttových diagramů.

Po detailním prozkoušení a aplikaci obou variant řešení (genetické algoritmy a lineární programování) jsem usoudila, že metoda genetických algoritmů není pro řešení složitých rozvrhovacích úloh příliš vhodná. Z použitých metod tedy hodnotím lineární programování jako metodu jednodušší a efektivnější pro řešení rozvrhovacích úloh. Uznávám však také, že je možné genetický algoritmus naprogramovat lépe tak, aby se s ním daly vyřešit i složitější příklady, napsání takového kódu je však dosti náročné a přesahuje rozsah diplomové práce. K tomuto tématu bych se však ráda vrátila v rámci mé disertační práce.

Závěr

Předmětem mé diplomové práce bylo zefektivnění výroby v nejmenované výrobní společnosti XY, konkrétně jsem se zabývala úpravou týdenního výrobního plánu uvedené organizace, prostřednictvím řazení zakázek na vstupu do systému.

V první části práce jsem důkladně prostudovala a popsala základy procesního řízení a reengineeringu operací v organizacích. Součástí procesního přístupu je také samotné řízení výroby a její optimalizace. Dále jsem již přistoupila ke konkrétní problematice rozvrhování výrobních plánů v zakázkové výrobě a prioritním pravidlům při tvorbě plánů. Rozvrhovací problém ve výrobě se řadí mezi kombinatorické úlohy, které je možné řešit hned několika metodami. Rozdělením a popisem jednotlivých metod, jak přesných, tak heuristických, jsem se zabývala v další části práce.

Důležitou součástí je metodika práce, ve které jsem se zabývala jednak specializovanými softwary a společnostmi, které se zabývají modelováním výroby a jejím zefektivňováním, zejména právě rozvrhováním zakázek ve výrobě a přiřazováním zaměstnanců k jednotlivým pracovištím. Hlavním smyslem metodické části je volba metod, které jsem následně použila pro řešení zadané úlohy. Pro vypracování výrobního plánu z konkrétních dat nasbíraných ve výrobním provozu společnosti XY, jsem si zvolila metodu genetických algoritmů a metodu lineárního programování.

V praktické části práce v první řadě popisuji získaná data, která jsem mírně přizpůsobila tak, aby se na ně daly zvolené metody rozumně aplikovat. Následně jsem v programovacím jazyku R nadefinovala zadanou úlohu tak, aby za pomoci genetických algoritmů bylo vygenerováno několik možných scénářů přípustných řešení a z nich poté vybráno, za dané situace, nejlepší možné suboptimální řešení. Podobně jako v tomto případě jsem se posléze upravila kód, pro řešení úloh projektového managementu za pomoci metody lineárního programování tak, aby fungoval i na mé zadání úlohy. K aplikaci metody lineárního programování jsem si zvolila software MATLAB, což je sofistikovaný počítačový program pro řešení rozsáhlých nejen matematických úloh a jejich modelování.

Po implementaci obou zvolených metod a následnému zakreslení výsledků do Ganttových diagramů jsem dospěla k závěru, že metoda lineárního programování je pro řešení konkrétní rozvrhovací úlohy, a možná i celkově kombinatorických úloh, vhodnější než metoda genetických algoritmů. Tímto vyvracím můj předpoklad, který jsem uvedla již na začátku práce, a sice, že metoda genetických algoritmů bude dosahovat lepších výsledků než metoda lineárního programování. Aplikace genetických algoritmů je příliš složitá a bohužel se mi nepodařilo zcela optimálně nastavit algoritmus tak, aby při výběru z počtu vygenerovaných řešení respektoval všechna

potřebná omezení. Metoda lineárního programování tedy v tomto případě fungovala lépe. Došla jsem však k závěru, že tato metoda je vhodná spíše pro řešení snadnějších a víceméně unifikovaných problémů. V reálné organizaci a jejím výrobním středisku vstupují do rozvrhu výrobních činností četné výjimky, nepředvídatelné situace, a také časy pro seřízení strojů, je tedy třeba výsledný plán vždy ještě ručně upravit.

Výsledkem mé práce je tedy praktická aplikace zvolených metod a navržení zakódování zadaného problému v profesionálních softwarech, jakými jsou R a MATLAB. Navržené algoritmy jsou po provedení určitých úprav vhodné pro aplikaci i pro jiné výrobní společnosti, řešící problematiku rozvrhování výroby. Cílem práce také bylo porovnat zvolené metody, při němž jsem zjistila, že metoda lineárního programování je pro takovouto aplikaci vhodnější, obě dvě metody však dosahují lepších výsledků než původně využívaná metoda ve společnosti XY, first come, first next.

Hlubší proniknutí do pokročilých rozhodovacích metod, jakými jsou genetické algoritmy a lineární programování bylo pro mě velmi přínosné a zajímavé. Práce s profesionálními a v praxi častěji využívanými programy, jakýmiž jsou MATLAB a R mi dala hodně zkušeností, které posléze určitě využiji, ať už v mém dalším studiu nebo v praxi. Myslím, že výsledky mé práce mohou být, do jisté míry, přínosné i pro výrobní společnost XY, která postupy a mé doporučení bude částečně implementovat do výroby. Pro detailnější vyhodnocení a zpracování její výrobní situace bych však doporučovala konzultaci s jednou ze specializovaných společností, které zmiňuji v metodické části práce.

Seznam literatury

- About Tomlab (2016) *tomopt.com*. Available at: <http://tomopt.com/tomlab/about/>.
- ABU-SRHAHN, A. 'a and AL HASAN, M. (2015) 'Hybrid Algorithm using Genetic Algorithm and Cuckoo Search Algorithm for Job Shop Scheduling Problem', *IJCSI International Journal of Computer Science Issues*, 12(2).
- AGRAWAL, R., PATTANAIK, L. N. and KUMAR, S. (2012) 'Scheduling of a flexible job-shop using a multi-objective genetic algorithm', *Journal of Advances in Management Research*, 9(2), pp. 178–188. doi: 10.1108/097279812112711922.
- Batch and Jobshop Production (2014) *managementguru*. Available at: <http://www.managementguru.net/batch-and-jobshop-production/>.
- BIERWIRTH, C. and MATTFELD, D. C. (1999) 'Production Scheduling and Rescheduling with Genetic Algorithms', *Evolutionary Computation*, 7(1).
- BIN TAIP, H., BIN ANUAR, M. S., BINTI KHAMIS, N. A. and AL SUNDARAM, P. (no date) 'Scheduling'. Available at: <http://slideplayer.com/slide/9329608/>.
- VOM BROCKE, J. and ROSEMAN, M. (2015) *Handbook on business process management 1*. 2nd edn. Available at: http://otgo.tehran.ir/Portals/0/pdf/Handbook%20on%20Business%20Process%20Management%201_1.pdf.
- BRUCKER, P. (1995) *Scheduling algorithms*. 5th edn. Berlin: Springer. Available at: http://www.math.nsc.ru/LBRT/k5/Scheduling/BruckerSchedulingAlgorithms_Full.pdf.
- BRYCHTA, K. (2008) *Vybrané kapitoly z operační analýzy*. 1st edn. Bučovice.
- CHASE, R. and AQUILANO, N. (1995) *Production and operation management: manufacturing and services*. 7th edn.
- CHEUNG, W. and ZHOU, H. (2001) 'Using Genetic Algorithms and Heuristics for Job Shop Scheduling with Sequence-Dependent Setup Times*', *Annals of Operations Research*, 107, pp. 65–81.
- CHIPPERFIELD, A., FLEMING, P., POHLHEIM, H. and FONSECA, C. (no date) 'Genetic Algorithm TOOLBOX for Use with MATLAB'.
- CHRYSSOLOURIS, G. and SUBRAMANIAM, V. (2001) 'Dynamic scheduling of manufacturing job shops using genetic algorithm', *Jurnal of Intelligent Manufacturing*. (12), pp. 281–293.

Co-když analýza (no date) managementmania. Available

at: <https://managementmania.com/cs/co-kdyz-analyza-what-if-analysis>.

DAVENPORT, T. H. and STODDARD, D. B. (1994) 'Reengineering: business change of mythic proportions?', *MIS Quarterly*, 18.2, pp. 121–127. doi: 10.2307/249760.

DELLA-CROCE, F., TADEIS, R. and VOLTA, G. (1995) 'Genetic Algorithm for the job shop problem'. (Pergamon).

DE SMET, G. (no date) *Job Shop Scheduling - class project -advice on references/alg. to use for implementing & getting experimental results*. Available

at: <http://stackoverflow.com/questions/13559219/job-shop-scheduling-class-project-advice-on-references-alg-to-use-for-implement>.

DORNDORF, U. and PESCH, E. (1995) 'Evolution based learning in a job shop scheduling environment'. (Pergamon), 22, pp. 25–40.

DOSTÁL, P. (2008) *Pokročilé metody analýz a modelování v podnikatelství a veřejné správě*.

DOSTÁL, P., REIS, K. and SOJKA, Z. (2005) *Pokročilé metody manažerského rozhodování: konkrétní příklady využití metod v praxi*. 1st edn.

DRDOVÁ, V. (2007) *Optimalizace výrobní linky ve vybraném výrobním podniku*. diplomová práce. Vysoká škola ekonomická v Praze. Available

at: <file:///C:/Users/Blanka/Downloads/2007-drdova-vendula-KMPH-760225.pdf>.

ExpertBA (2012) *Business process management*. Available

at: <http://expertbusinessanalyst.com/business-process-management/>.

FERGUSON, T. S. (2015) 'Linear Programming'. Available

at: <https://www.math.ucla.edu/~tom/LP.pdf>.

FRIEBELOVÁ, J. (2006) 'Lineární programování'. Available

at: http://www2.ef.jcu.cz/~jfrieb/rmp/data/teorie_oa/LINEARNI_PROGRAMOVANI.pdf.

GONCALVES, J. F., MAGALHAES MENDES, J. and RESENDE, M. (2002) 'A Hybrid Genetic Algorithm for the Job Shop Scheduling Problem'.

GRAHAM, R. L., LAWLER, E. L., LENSTRA, J. K. and RINNOOY KAN, A. H. G. (1979) 'Optimization and approximation in deterministic sequencing and scheduling: a survey', *Annals of discrete mathematics*, 5, pp. 287–326. Available

at: http://www.math.ucsd.edu/~ronspubs/79_03_scheduling_survey.pdf.

GRASSEOVÁ, M., DUBEC, R. and HORÁK, R. (2008) *Procesní řízení ve veřejném i soukromém sektoru*. 1st edn. Brno: Computer Press, a.s.

- GUBÁN, M. and GUBÁN, Á. (2012) 'Production scheduling with genetic algorithm', *Advanced logistics systems*, 6(1), pp. 33–44.
- GUPTA, A. (2014) 'Scheduling and sequencing'. Available at: <https://www.slideshare.net/akankshagupta963871/scheduling-and-sequencing>.
- HAQ, A. N., BALASUBRAMANIAN, K., SASHIDHARAN, B. and KARTHICK, R. B. (2004) 'Parallel line job shop scheduling using genetic algorithm', *Int J Adv Manuf Technol*. doi: 10.1007/s00170-007-1164-z.
- HAMMER, M. and CHAMPY, J. (2009) *Reengineering the Corporation: A Manifesto for Business Revolution*.
- HANZAL, M. (2014) *Genetický algoritmus jako metoda řešení rozvrhovací úlohy*. bakalářská práce. Vysoká škola ekonomická v Praze.
- HART, P. (2008) *Mravenčí kolonie*. diplomová práce. Vysoké učení technické v Brně. Available at: https://dspace.vutbr.cz/bitstream/handle/11012/6814/Diplomka_alfa.pdf?sequence=1&isAllowed=y.
- HAZIZA, F. (2007) 'Scheduling Algorithms'. Uppsala University. Available at: https://www.it.uu.se/edu/course/homepage/oskomp/vt07/lectures/scheduling_algorithms/handout.pdf.
- HEŘMAN, J. (2001) *Řízení výroby*. 1st edn.
- HLINĚNÝ, P. (2007) 'Úloha lineární optimalizace'. Available at: <http://www.fi.muni.cz/~hlineny/Teaching/OU/OU-lect--3.pdf>.
- HYNEK, J. (2008) *Genetické algoritmy a genetické programování*. 1st edn.
- JABLONSKÝ, J. (2007) *Operační výzkum*. 3. edn.
- JAIN, A. S. and MEERAN, S. (1998) 'Deterministic job-shop scheduling: past, present and future', *Department of Applied Physics and Electronic and Mechanical Engineering*. Available at: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.9979&rep=rep1&type=pdf>.
- Job Shop Scheduling* (2017) wikipedia. Available at: https://en.wikipedia.org/wiki/Job_shop_scheduling.
- KALČEVOVÁ, J. (2017) 'Opakování lineárního programování'. Available at: <http://jana.kalcev.cz/vyuka/kestazeni/4EK311-cv05-reseni.pdf>.

KLICNAROVÁ, J. (2006) 'Lineární programování'. Available at: <http://www2.ef.jcu.cz/~janaklic/emm/predn4.pdf>.

KOBLASA, F. (no date) 'Rozvrhování výroby'. Liberec.

KOLÁČEK, J. and KONEČNÁ, K. (2014) 'Jak pracovat s MATLABem'. Available at: <https://www.math.muni.cz/~kolacek/vyuka/vypsyst/navod.pdf>.

KOLHARKAR, S. and ZANWAR, D. R. (2013) "Scheduling in Job Shop Process Industry", *IOSR Journal of Mechanical and Civil Engineering*, 5. Available at: <http://iosrjournals.org/iosr-jmce/papers/vol5-issue1/A0510117.pdf?id=2490>.

KVASNIČKA, V., POSPÍCHAL, J. and TIŇO, P. (2000) *Evolučné algoritmy*. Available at: <http://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Bool Evolutionary%20Algorithms.pdf>.

KWOK, Y.-K. and AHMAD, I. (1999) 'Static scheduling algorithms for allocating directed task graphs to multiprocessors', *Journal ACM Computing Surveys*, 31(4), pp. 406–471. Available at: http://delivery.acm.org/zdroje.vse.cz/10.1145/350000/344618/p406-kwok.pdf?ip=146.102.19.70&id=344618&acc=ACTIVE%20SERVICE&key=D6C3EEB3AD96C931%2EF07030D8BB94E8BC%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&CFID=734974626&CFTOKEN=89621750&_acm_=1488624170_dca24dd6741089a287111c25013e886e.

LAGOVÁ, M. and JABLONSKÝ, J. (1999) *Lineární modely*. Praha.

VAN LAARHOVEN, P. T. M., AARTS, E. H. L. and LENSTRA, J. K. (1990) 'Job shop scheduling by simulated annealing', *Institute for Operations Research and the Management Sciences*. (University of Technology, Eindhoven). Available at: <https://app.cs.amherst.edu/~ccmcgeoch/cs34/papers/jobshop171189.pdf>.

LHOTÁK, R. (2010) *Plánování a optimalizace zakázkové výroby v systému AROP*. Available at: <http://cvis.cz/hlavni.php?stranka=novinky/clanek.php>.

LINEÁRNÍ PROGRAMOVÁNÍ A ŘEŠENÍ NĚKTERÝCH ÚLOH SIMPLEXOVOU METODOU (2012). Available at: <https://aztli.wordpress.com/2012/02/09/linearni-programovani-a-reseni-nekterych-uloh-simplexovou-metodou/>.

'Lineární programování' (no date). Available at: <http://nb.vse.cz/~fabry/Graficke-reseni.pdf>.

MACH, M. (2009) *Elovučné algoritmy: Prvky a princípy*. Košice. Available at: <http://neuron-ai.tuke.sk/machm/publications/kniha-eapp.pdf>.

- MAJER, P. (2003) *Moderní metody rozvrhování výroby*. Ph.D. thesis. Vysoké učení technické v Brně.
- The MathWorks, Inc. (2017) *MATLAB*. Available at: <https://www.mathworks.com/products/matlab.html>.
- MATOUŠEK, J. (2006) 'Lineární programování'. Available at: <http://iti.mff.cuni.cz/series/2006/311.pdf>.
- MENLIG, F. and KOBLASA, F. (2015) 'Rozvrhování a plánování výroby'. Available at: https://www.unipranet.zcu.cz/doc/15_Planovani%20a%20rozvrhovani%20vyroby%20Manlig%20&%20Koblasa%20TUL.pdf.
- MERICA S.R.O. (2004) *Optimalizace výroby*. Available at: <http://merica.cz/cs/services/productionoptimization/flowshopwithblocking/>.
- MERZ (2013) *Případová studie TRW*. Available at: <http://www.merz.cz/reference/reference-detail/r/pripadova-studie-trw>.
- MINAŘÍK, J. (2007) *Evoluční algoritmy-MATLAB Gate Toolbox*. diplomová práce. Vysoké učení technické v Brně. Available at: http://autnt.fme.vutbr.cz/szz/2007/DP_Minarik.pdf.
- MONTMIRAIL, V. (2015) *Flow/Job Shop to Boolean satisfiability [Polynomial-time reduction] part 2*. Available at: <http://stackoverflow.com/questions/29651856/flow-job-shop-to-boolean-satisfiability-polynomial-time-reduction-part-2>.
- MOTALÍK, P. (2009) *Procesní řízení a jeho optimalizace v organizaci*. diplomová práce. Bankovní institut vysoká škola Praha. Available at: https://is.bivs.cz/th/10726/bivs_m/DP_PetrMotalik.pdf.
- Operating System Scheduling algorithms* (2017) *tutorialspoint.com*. Available at: https://www.tutorialspoint.com/operating_system/os_process_scheduling_algorithms.htm.
- PAPPOVÁ, B. (2010) *Optimalizace výrobních procesů*. Vysoké učení technické v Brně. Available at: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=28647.
- PEŠKOVÁ, S. (2016) 'Schéma procesu'. Available at: <http://slideplayer.cz/slide/3654451/>.
- PHANDEN, R. K., JAIN, A. and VERMA, R. (2012) 'A genetic algorithm-based approach for job shop scheduling', *Journal of Manufacturing Technology Management*, 23(7), pp. 937–946.

Planning And Control For Job Shop Production (2014) *tutorsonnet*. Available at: <http://www.tutorsonnet.com/planning-and-control-job-shop-production-homework-help.php>.

Podnikový proces (2016) *managementmania*. Available at: <https://managementmania.com/cs/business-process-podnikovy-proces>.

Procesní řízení (2016) *wikipedia*. Available at: https://cs.wikipedia.org/wiki/Procesn%C3%AD_%C5%99%C3%ADzen%C3%AD.

Process approach (1999). Available at: <https://www.pqbweb.eu/platform.php?i=&if=27&ch=384>.

SADEGHEIH, A. (2006) 'Scheduling problem using genetic algorithm, simulated annealing and the effects of parameter values on GA performance', *Elsevier*. Available at: http://ac.els-cdn.com/S0307904X05000521/1-s2.0-S0307904X05000521-main.pdf?_tid=cb1f1324-fb40-11e6-8761-00000aacb360&acdnat=1488016818_fc39e041aa852cb31f5bc3ace05d1d73.

SCHAUDER, J. (no date) *Flow Shop to Boolean satisfiability [Polynomial-time reduction]*. Available at: <http://stackoverflow.com/questions/29366185/flow-shop-to-boolean-satisfiability-polynomial-time-reduction>.

Scheduling (production processes) (no date) *wikipedia*. Available at: [https://en.wikipedia.org/wiki/Scheduling_\(production_processes\)](https://en.wikipedia.org/wiki/Scheduling_(production_processes)).

Simplexová metoda (2016) *algoritmy.net*. Available at: <https://www.algoritmy.net/article/1416/Simplexova-metoda>.

S, T. (no date) *Job scheduling problem*. Available at: <http://stackoverflow.com/questions/1932545/job-scheduling-problem>.

SVOBODOVÁ, H. (2008) *Produkční a operační management*. Available at: https://www.vsem.cz/data/data/sis-ukazky-kapitol/POM_Ukazka_kapitoly.pdf.

ŠKRABAL, O. (2010) *Genetické algoritmy a rozvrhování*. Vysoké učení technické v Brně.

ŠMEREK, M. and MOUČKA, J. (2008) '*Ekonomicko-matematické modely*'. Available at: http://mendelu.org/upload/skripta_k_predmetu.pdf.

TALBI, E.-G. and FAROUK, Y. (2016) *Metaheuristics for production systems*. Edited by L. Amodio. Springer. Available at: <https://books.google.cz/books?id=NcLCwAAQBAJ&pg=PA323&lpg=PA323&dq=Optimal+Fuzzy+counterparts+of+scheduling+rules&source=bl&ots=bMmfq2q0zN&sig=kJEfd84jJfCr3y1EFcPtr2cbJDA&hl=cs&sa=X>

&ved=0ahUKE-wjR07TG6avSAhXGhywKHX27AfcQ6AEIMjAC#v=onepage&q=Optimal%20Fuzzy%20counterparts%20of%20scheduling%20rules&f=false

TIAN, X. and WANG, J. (2012) 'The Simulation Optimization for Job-Shop Scheduling Based on Plant Simulation Using Genetic Algorithm', *Applied Mechanics and Materials*, pp. 217–219. doi: 10.4028/www.scientific.net/AMM.217-219.1444.

TOMLAB models manual (2002) *TOMLAB optimization*. Available at: http://tomopt.com/docs/models/tomlab_models024.php#htoc50.

TSAI, J.-T., LIU, T.-K., HO, W.-H. and CHOU, J.-H. (2007) 'An improved genetic algorithm for job-shop scheduling problems using Taguchi-based crossover', *Int J Adv Manuf Technol*.

'Úlohy rozvrhování a plánování' (no date). Available at: <http://labe.felk.cvut.cz/~tkrajnik/sdu/slides/slides10.pdf>.

'Úvod do lineárního programování' (no date). Available at: <http://labe.felk.cvut.cz/~tkrajnik/sdu/slides/slides8.pdf>.

VANDDERBEI, R. J. (2006) *Linear Programming: Foundations and Extensions*. 2nd edn. Princeton. Available at: https://support.dce.felk.cvut.cz/pub/hanzalek/_private/ref/Vanderbei_Linear_Programming.pdf.

VANĚČKOVÁ, E. (1996) *Ekonomicko-matematické metody*. České Budějovice.

WANER, F. (no date) *Genetic Algorithms for Shop Scheduling problems: a survey*. a survey. Germany: Otto-von-Guericke-Universitat.

What is R? (no date) www.r-project.org. Available at: <https://www.r-project.org/about.html>.

YAMADA, T. and NAKANO, R. (1997) 'Genetic Algorithms for Job-Shop Scheduling Problems', *Modern Heuristic for Decision Support*, UNICOM seminar, pp. 67–81.

ZELENÁ, V. (2015) 'Znalostní management'. Jindřichův Hradec.

ZIGIARIS, S. (2000) 'Business process re-engineering'. Available at: http://www.adi.pt/docs/innoregio_BPR-en.pdf.

Přílohy

Příloha 1-Původní zdrojová data

Deadline	Přířazený ná- zev	Pracovní název-původní	Počet ks	Ope- race1	Ope- race2	Ope- race3	Ope- race4	Ope- race5	Ope- race6	Cel- kem
50 J1	výlisek_01	240.09K02	6000	5	2	6				13
50 J2	výlisek_02	240.40K02	12000	7	13					19
50 J2	výlisek_02	240.40K02	12000	7	13					19
1 J3	výlisek_03	220.16K02	10000	7	8	12	8			36
3 J2	výlisek_02	240.40K02	12000	7	13					19
3 J4	výlisek_04	116.12/9	30000	17	5	5				27
45 J5	kovovydil_01	0534_119	30000	4	4	3	2	5	4	22
46 J5	kovovydil_01	0534_119	30000	4	4	3	2	5	4	22
47 J5	kovovydil_01	0534_119	30000	4	4	3	2	5	4	22
48 J5	kovovydil_01	0534_119	30000	4	4	3	2	5	4	22
49 J5	kovovydil_01	0534_119	30000	4	4	3	2	5	4	22
30 J6	kovovydil_02	0476_119 K02	10000	13	8	5				26
objednávka	J7	výztuha	výztuhy 476.119	50920	7	8	5			20
46 J8	atyp_01	Metal Insert MBA0289-A01-06 ZE-5097	70000	9	5	8	6			27
49 J9	kovovydil_z	Clamp 1900987	2000	25	10					35
49 J8	atyp_01	Metal Insert MBA0289-A01-06 ZE-5097	70000	9	5	8	6			27
50 J8	atyp_01	Metal Insert MBA0289-A01-06 ZE-5097	60000	8	4	6	5			23
2 J8	atyp_01	Metal Insert MBA0289-A01-06 ZE-5097	70000	9	5	8	6			27

			Metal Insert MBA0289-A01-06 ZE-5097	70000	9	5	8	6			27
3	J8	atyp_01									
pojistná zásoba	J10	výlisek_05	04995-výlisek-Stiffener	1500	3	12					14
pojistná zásoba	J11	panel_x	04988-58213-CTR Floor panel,RH	2000	2	3	2				6
pojistná zásoba	J12	panel_y	04989-58214-CTR Floor panel,LR	2000	2	2	2	1			7
44	J12	panel_y	04989-58214-CTR Floor panel,LR	5000	4	6	5	3			17
44	J11	panel_x	04988-58213-CTR Floor panel,RH	5000	4	6	4				15
45	J12	panel_y	04989-58214-CTR Floor panel,LR	5000	4	6	5	3			17
45	J11	panel_x	04988-58213-CTR Floor panel,RH	5000	4	6	4				15
42	J13	atyp_02	STE00M5018	200	3	3	7	7			19
42	J14	atyp_03	STD003M5018	1500	2	3	8	1	30		44
48	J15	atyp_04	ST7020M5005	1000	14	7					21
47	J16	clip	KICKDOWN CLIP	22000	10	6	3				19
47	J17	páka_01	Páka BMW 26.24.	6000	5	3	5				12
48	J18	atyp_05	SPRING-EXT,ETC KD,USPRING	80000	9	4	5	8			26
49	J17	páka_01	Páka BMW 26.24.	6000	5	3	5				12
1	J17	páka_01	Páka BMW 26.24.	6000	5	3	5				12
2	J17	páka_01	Páka BMW 26.24.	6000	5	3	5				12
2	J16	clip	KICKDOWN CLIP	22000	10	6	3				19
3	J17	páka_01	Páka BMW 26.24.	6000	5	3	5				12
5	J17	páka_01	Páka BMW 26.24.	6000	5	3	5				12
5	J16	clip	KICKDOWN CLIP	22000	10	6	3				19
49	J19	atyp_06	Mi-40x2,5-M8x35	2000	1	4	9	3	2		18
50	J20	atyp_07	Mi-30x2,5-M8x20	13000	26	3	3	13			45
Sklad	J21	kovovydil_03	MQP 21-72	130	0	0	1	0	0	2	4
Sklad	J22	atyp_08	MQP 82	8000	1	5	9				15

Sklad	J23	atyp_09	MVA-L	16000	13	5					18
Sklad	J24	výlisek_06	Aufhangebugel	4800	2	10	10	5			27
Sklad	J25	výlisek_07	Gerustschuh 9kN	2000	1	2	6	7	3		18
46	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	4	9
47	J25	výlisek_07	Gerustschuh 9kN	2880	2	2	8	10	4		25
47	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	4	9
47	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	4	9
48	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	4	9
48	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	4	9
48	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	4	9
49	J21	kovovydil_03	MQP 21-72	150	0	0	2	0	1	2	5
49	J21	kovovydil_03	MQP 21-72	300	1	0	3	0	1	1	6
50	J26	kovovydil_04	AC 0400.2150.0000 bracket S-1,5	1000	1	1	2	4	3		11
1	J26	kovovydil_04	AC 0400.2150.0000 bracket S-1,5	1000	1	1	2	4	3		11
5	J26	kovovydil_04	AC 0400.2150.0000 bracket S-1,5	2000	2	1	4	9	7		23
52	J26	kovovydil_04	AC 0400.2150.0000 bracket S-1,5	2300	2	2	5	10	8		26
48	J27	kovovydil_05	AC 0400.2254.0000 bracket S-1,5	900	0	0	2	2			4
50	J27	kovovydil_05	AC 0400.2254.0000 bracket S-1,5	1000	0	1	2	2			5
3	J27	kovovydil_05	AC 0400.2254.0000 bracket S-1,5	600	0	0	1	1			3
52	J27	kovovydil_05	AC 0400.2254.0000 bracket S-1,5	5000	1	3	10	10			23
Celkem											1129,5

